Linguistic Knowledge Acquisition from Parsing Failures

Masaki KIYONO* and **Jun-ichi TSUJII**
(kiyono@ccl.umist.ac.uk and tsujii@ccl.umist.ac.uk)
Centre for Computational Linguistics
University of Manchester Institute of Science and Technology
PO Box 88, Manchester M60 1QD
United Kingdom

Abstract

A semi-automatic procedure of linguistic knowledge acquisition is proposed, which combines corpus-based techniques with the conventional rule-based approach. The rule-based component generates all the possible hypotheses of *defects* which the existing linguistic knowledge might contain, when it fails to parse a sentence. The rule-based component does not try to identify the defects, but generates a set of hypotheses and the corpus-based component chooses the plausible ones among them. The procedure will be used for adapting or re-using existing linguistic resources for new application domains.

1 Introduction

While quite a number of useful grammar formalisms for natural language processing now exist, it still remains a time-consuming and hard task to develop grammars and dictionaries with comprehensive coverage. It is also the case that, though quite a few computational grammars and dictionaries with comprehensive coverage have been used in various application systems, to re-use them for other application domains is not always so easy, even if we use the same formalisms and programs such as parsers, etc. We usually have to revise, add, and delete grammar rules and lexical entries in order to adapt them to the peculiarities of languages (sublanguages) of new application domains [Sekine *et al.*, 1992; Tsujii *et al.*, 1992; Ananiadou, 1990].

*also a staff member of Matsushita Electric Industrial Co.,Ltd., Tokyo, JAPAN.

Such adaptations of existing linguistic knowledge to a new domain are currently performed through rather undisciplined, trial and error processes involving much human effort. In this paper we show that techniques similar to those in robust parsing of ill-formed input, together with corpus-based techniques, can be used to discover disparities between existing linguistic knowledge and actual language usage in a new domain, and to hypothesize new grammar rules or lexical descriptions.

Although our framework appears similar to grammar learning from corpora, our current goal is far more modest, i.e. to help linguists revise existing grammars by showing possible defects and hypothesizing them through corpus analysis.

2 Robust Parsing and Linguistic Knowledge Acquisition

2.1 Search Space of Possible Hypotheses

When a parser fails to analyse an input sentence, a robust parser hypothesizes possible errors in the input in order to complete the analysis and correct errors [Douglas and Dale, 1992]: for example, deletion of necessary words (Ex. I have book), insertion of unnecessary words (Ex. I have a the book), disorder of words (Ex. I a book have), spelling errors (Ex. I have a bok), etc.

As there is usually a set of possible hypotheses to complete the analysis, this error detection process becomes non-deterministic. Furthermore, allowing operations such as deletion and insertion of arbitrary sequences of words or unrestricted permutation of word sequences, radically expands its search space. The process generates many nonsensical hypotheses unless we restrict the search space either by heuristics-based cost functions [Mellish, 1989], or

Type of Failures	Robust Parsing	Knowledge Acquisition
Remaining Constituents to be Collected	hypotheses of - deletion of necessary words - insertion of unnecessary words - disorder of words	hypotheses of - lack of necessary rules
Failure of Application of an Existing Rule	relaxation of - feature agreements	identification of - disagreeing features
Unrecognized Sequence of Characters	hypotheses of - spelling errors	hypotheses of - new words

Table 1: Types of Hypotheses

by introducing prior knowledge about regularities of errors in the form of annotated rules [Goeser, 1992].

On the other hand, our framework of ***knowledge acquisition from parsing failures*** does not assume that the input contains errors, but instead, assumes that linguistic knowledge of the system is incomplete. This means that we do not need to, or should not, allow the costly operations of changing input, and therefore the search space explosion encountered by a robust parser does not occur.

For example, when a string of characters which is not registered in the dictionary as a word appears, a robust parser may assume that there are spelling errors and try to identify the errors by changing the character string (deleting characters, adding new characters, etc.) to find the "closest" legitimate word in the dictionary. This is because the dictionary is assumed to be complete, e.g. that it contains all lexical items that will appear. On the other hand, we simply hypothesize that the string of characters is a word which should be registered in the dictionary, together with the lexical properties that are compatible with those hypothesized from the surrounding syntactic/semantic context in the input.

Table 1 shows different types of hypotheses to be produced by a robust parser and a program for knowledge acquisition from parsing failures.

Although the assumption of legitimacy of input reduces significantly the size of the search space, the assumption of incomplete linguistic knowledge introduces another type of non-determinism and potentially a very large search space. For example, even if a word is registered in the dictionary as a noun, it can have in theory arbitrary parts of speech such as verb, adjective, adverb, etc., as there is no guarantee that the current dictionary exhausts all possible usages of the word. A simple method will end up with an explosion of hypotheses.

2.2 Corpus-based Knowledge Acquisition

Apart from the differences in types of hypotheses, an essential difference exists in the very nature of errors in the two paradigms. While errors in ill-formed input, by definition, are supposed not to show any significant regularity incompleteness or "linguistic knowledge errors" are supposed to be observed recurrently in a corpus.

¿From the practical viewpoint of adaptation of knowledge to a new application domain, disparities between existing knowledge and actual language usages which are manifested only rarely in a reasonable size sample corpus, are less significant than those recurrently observed. Furthermore, unlike robust parsing, we do not need to identify causes of parsing failures at the time of parsing. That is, though there is in general a set of hypotheses which equally explain parsing failures of single sentences, we can choose the most plausible ones by observing statistical properties (for example, frequencies) of the same hypotheses generated in the analysis of a whole corpus. This would be a reasonable approach, as significant disparities between knowledge and actual usages are supposed to be observed recurrently.

One of the crucial differences between the two paradigms, therefore, is that unlike robust parsing, we need not narrow down the number of hypotheses to one by using heuristics based on cues inside single sentences. Multiple hypotheses are not seriously damaging, though it is desirable for them to be reasonably restricted. The final decision will be made through the observation of hypotheses generated from the analysis of a whole corpus.

3 Formalism and the Parser

3.1 Linguistic Knowledge to be Acquired

The formalism and linguistic theories which one chooses as the bases for grammatical learning largely determine the types of linguistic knowledge to be acquired as well as their representational forms.

If one chooses a general form of CFG without committment to any specific linguistic theory, the knowledge to be learned is just a set of general rewriting rules. On the other hand, if one chooses more specific linguistic frameworks, they impose further restrictions on possible forms of knowledge to be learned, and introduce more diverse forms of representing knowledge. For example, if one chooses a lexicon-oriented framework, it may assume the existence of *subcategorization frames* as lexical properties, and impose restrictions on the form of rewriting rules such as "the LHS of each rewriting rule should

Rewriting Rule:

$Cat(F) \Rightarrow Cat1(F1) + Cat2(F2) + \cdots + Catn(Fn) : f(F, F1, F2, \ldots, Fn).$

Lexical Rule:

$Cat(F) \Rightarrow [Word1, Word2, \ldots, Wordn] : f(F).$

Figure 1: General Forms of Grammar Rules

have one and only one head", etc.

While minimal commitment to specific linguistic theories is possible for research on general algorithms of robust parsing (as in [Mellish, 1989]), it does not seem feasible for our paradigm, as our aim (learning linguistic knowledge) is directly related to the problems of what type of knowledge is to be learned and how it is properly represented. To learn such *meta-principles* from corpora, starting from a weak assumption formalism like CFG, requires induction and an impractically huge search space.

Instead, our aim is far less ambitious than automatic grammar learning from corpora. Our goal is to make existing grammar and lexical resources more comprehensive or to adapt them to new application domains. That is, from the very beginning, a system has a set of linguistic knowledge represented in specific forms by assuming that meta-principles proposed by current linguistic theories are valid. We use established linguistic concepts such as 'Number-Property', subcategorization frames of predicates, syntactic categories, etc. Most of the inductive processes required in grammar learning will have been performed in advance (by linguists), though hypothesizing lacking knowledge may require induction even in our framework.

3.2 Grammar Formalism

Figure 1 and Figure 2 show the general forms of the rules in our grammar and specific examples respectively. For experiments, we use a grammar which consists of 190 rewriting rules, giving us reasonable coverage of English.

As can be seen, the formalism used is a conventional kind of unification grammar where context free rules are augmented by feature conditions. In Figure 1, each syntactic category $Cati$ in a rewriting rule has a feature structure F_i, which is unified either wholly or partially to another by using the same variable or by applying the unification function $f(F, F_1, F_2, \ldots, F_n)$ (See examples in Figure 2).

Although we do not commit ourselves to any specific linguistic theory, it can be seen from the example rules that we use basic concepts in modern linguistic theories such as Head, Subcat, a set of grammatical functions (Subject, Object, etc.), etc.

```
s(F) ⇒ np(F_np) + vp(F_vp) :
                  (head,F) = (head,F_vp),
                  (first,subcat,F_vp) = F_np.

vp(F) ⇒ vp(F_vp) + np(F_np) :
                  (head,F) = (head,F_vp),
                  (subcat,F) = (rest,subcat,F_vp),
                  (first,subcat,F_vp) = F_np.

v(F) ⇒ [has] :
       (pred,head,F) = have,
       (obj,head,F) = (head,first,subcat,F),
       (subj,head,F) = (head,first,rest,subcat,F),
       (psn,subj,head,F) = 3,
       (nbr,subj,head,F) = sgl,
       (cat,first,subcat,F) = np,
       (cat,first,rest,subcat,F) = np.
```

Figure 2: Examples of Grammar Rules

3.3 Parsing Results

The parser we use is a left corner, bottom-up parser with top-down filtering. When it fails to parse, it reparses the same sentence without top-down filtering and outputs the following intermediate tuples.

Successful Category:
`successful_goal(Cat,Words,WordsRest)`

This tuple means that a word sequence between '`Words`' and '`WordsRest`' was successfully analysed as an expected category '`Cat`'.

ex.) successful_goal(np, [the,boy,has,a,book], [has,a,book])

Failed Category: `failed_goal(Cat,Words)`

This tuple means that an expected category '`Cat`' could not be analysed from a word list '`Words`'.

ex.) failed_goal(np,[has,a,book])

These tuples are similar to active and inactive edges of a chart parser but the 'Failed Category' above directly expresses the local ungrammaticality while an active edge expresses an incomplete expectation of a category within a grammar rule.

4 Generation of Hypotheses

4.1 Hypothesizing Grammar Rules from Parsing Failures

When the parser fails to analyse a sentence, the grammar rule hypothesizing program (shortly GRHP) investigates the parsing results and hypothesizes all the possible modifications of the existing grammar that produce a complete parsing result. GRHP starts from the top category '*s*' and proceeds by breaking down each failed category in accordance with the existing grammar.

The hypothesizing procedure (hypo_proc) works for each category $CatA$ as follows (See also Figure 3):

```
hypo_proc(CatA)
  begin
    if (CatA is a failed category) then
      foreach i (CatA ⇒ CatB_i1 + ··· + CatB_in)
                                        ····· (1)
        foreach j (CatB_ij)
          call hypo_proc(CatB_ij)
                                        ····· (2)
          if (CatB_ij is a failed category) then
            HYPO(left_recursive_rule(CatB_ij-1))
                                        ····· (3)
          endif
        end
        HYPO(feature_disagreement(B_i1, ..., B_in))
                                        ····· (4)
      end
    endif
    if (CatA is a non-lexical category) then
      HYPO(rule: CatA ⇒ CatC_1 + ··· + CatC_l)
                                        ····· (5)
    else if (CatA is a failed category) then
      HYPO(lexical_entry: CatA ⇒ [Word])
                                        ····· (6)
    endif
  end
```

(1) If $CatA$ is a failed category, the procedure breaks $CatA$ down into its daughter categories according to the rule '$CatA \Rightarrow CatB_{i1} + \cdots + CatB_{in}$' in the existing grammar. The procedure iterates this breakdown for each rule composing $CatA$.

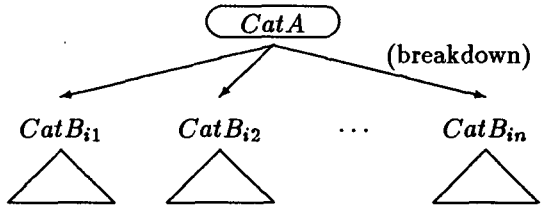
(2) The procedure calls itself recursively for each daughter category $CatB_{ij}$.

(3) The procedure also checks whether $CatB_{ij}$ is a failed category. If it is a failed category, the procedure hypothesizes a *new left recursive rule* for the preceding category $CatB_{ij-1}$ and generates a rule '$CatB_{ij-1} \Rightarrow CatB_{ij-1} + CatR_1 + \cdots + CatR_o$' by searching adjacent successful categories next to $CatB_{ij-1}$ unless this rule is included in the existing grammar.

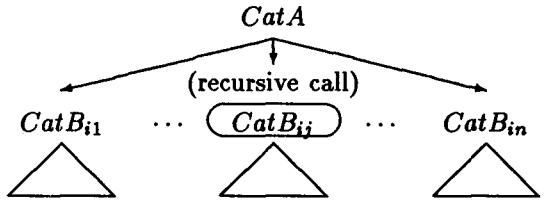
(4) If all the daughter categories are successful categories, the procedure hypothesizes the *feature disagreement* between them. For example, if the existing grammar contains a rule '$s \Rightarrow np + vp$' and both 'np' and 'vp' are successfully parsed but still 's' is a failed category, the procedure hypothesizes the feature disagreement between 'np' and 'vp'.

(5) When the procedure finishes applying all the known rules of $CatA$, it hypothesize a *new rule* of $CatA$ unless $CatA$ is a lexical category. The procedure searches adjacent successful categories starting from the word position where $CatA$ is expected and generates a rule

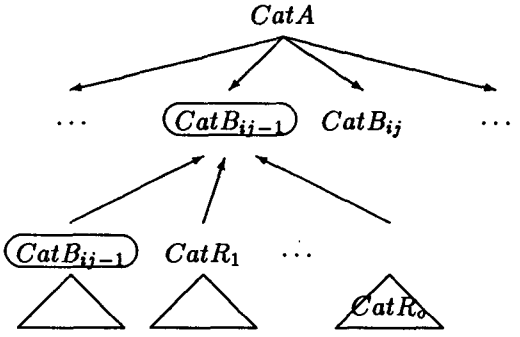
(1) Breakdown of a Failed Category



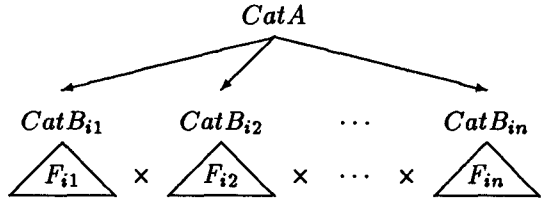
(2) Recursive Breakdown



(3) Hypothesizing a New Left Recursive Rule

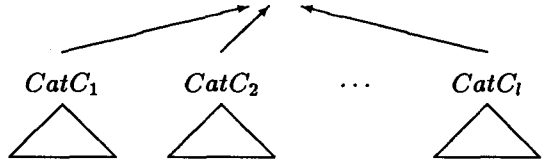


(4) Hypothesizing a Feature Disagreement



(5) Hypothesizing a New Rule

$CatA \Rightarrow CatC_1 + CatC_2 + \cdots + CatC_l$



(6) Hypothesizing a New Lexical Entry

$CatA \Rightarrow [Word]$

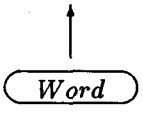


Figure 3: Hypothesizing Process

'$CatA \Rightarrow CatC_1 + \cdots + CatC_l$' unless the rule is included in the existing grammar. This step is directly executed if $CatA$ is not a failed category or there are no known rules which compose $CatA$.

(6) If $CatA$ is a failed lexical category, the procedure hypothesizes a *new lexical entry* '$CatA \Rightarrow [Word]$' at the word position where $CatA$ is expected. By this hypothesis, an unknown word as well as a known word is assigned into an expected category.

Actually, this process is implemented on Prolog and each hypothesis is generated alternatively. When GRHP generates a hypothesis, it passes the hypothesis to the parser to analyse the remaining part of the sentence. As the result, GRHP outputs only the hypotheses that lead to complete structures of the sentences.

On this search algorithm, we imposed a strict condition that a sentence does not have more than one cause of its parsing failure and the combination of hypotheses is not allowed to account for one ungrammaticality. Therefore, GRHP generates each hypothesis independently and all the hypotheses generated from a sentence are alternatives.

4.2 Elimination of Redundant Hypotheses

GRHP in Section 4.1 generates a lot of alternative hypotheses, many of which are nonsensical from the linguistic viewpoint. GRHP as it is stated there does not include any criteria for judging the appropriateness of hypotheses as linguistic rules. In the extreme, it can hypothesize a rule which directly derives the input string of words from the start symbol '*s*'. Although such a rule allows the grammar to accept the input as a sentence, the rule obviously lacks the *generality* which we expect a linguistic rule to have. More seriously, it ignores all the generalizations which the existing grammar embodies.

One can conceive of an automatic procedure of grammar learning which starts from a set of such rules and gradually discovers grammatical concepts, such as NP, VP, etc., based on the replaceability among sub-strings. However, as we discussed in Section 3, such a procedure has to solve the difficulties caused by a huge search space which an induction process generally has, and we are convinced that it is impossible to induce from scratch the rules involved in complex systems such as human languages.

Instead, our framework assumes that most of the induction processes required in grammar learning have been done by linguists and embodied in the form of the existing grammar. The system has only to discover defects or incompleteness of the existing grammar or to discover the differences between the *sublanguage* in a new domain and the *sublanguage* which the existing grammar has been prepared for. In other words, the hypotheses GRHP generates should use the generalizations embodied in the existing grammar as much as possible, and the hypotheses which ignore them should be rejected as nonsensical or redundant ones.

GRHP hypothesizes a set of new rules which collect sequences of successful categories starting at the same word position into the same failed category. If a substring of the input which is collected into the failed category contains a sequence of "a good student", for example, and if the existing grammar contains rules like '*nhead* $\Rightarrow$ *adj* + *nhead*', '*np* $\Rightarrow$ *det* + *nhead*', etc., GRHP will generate hypotheses whose RHSs contain the sequence, such as '*det* + *adj* + *nhead*', '*det* + *nhead*', etc., as well as the ones whose RHSs contain '*np*' for the same part of the input.

However, because the hypothesized rules containing smaller constituents, such as '*det*', '*nhead*', etc. instead of '*np*', ignore the generalization captured by '*np*' in the existing grammar, they should be disregarded as redundant, while only the ones which contain '*np*' in their RHSs are kept as viable hypotheses.

Much simpler criteria could also be used to prevent nonsensical hypotheses from being generated. For example, a rule whose RHS consists of a large number of constituents would not be viable, if we assume that the existing grammar has already been equipped with a reasonable set of syntactic categories (non-terminals) which allow sentences to be assigned reasonably *structured* descriptions.

The following is a list of the criteria which GRHP can use to disregard nonsensical hypotheses.

[1] **Priority to the hypotheses of feature disagreement:** Assuming that the existing grammar is quite comprehensive, we can give priority to the hypotheses of feature disagreement, which do not create new rules. In the current implementation, if GRHP finds a feature disagreement hypothesis to restore a failed category, it stops the recursion and generates no more hypotheses.

[2] **Number of daughter nodes:** A rule which collects an excessive number of constituents into one large constituent at once is not viable. We currently restrict the number of daughter nodes to 4.

[3] **Priority to the hypotheses using generalizations embodied by the existing grammar:** As discussed in the above, priority is given to the hypotheses which contain '*np*' as daughters over those which contain '*det* + *nhead*', '*det* + *adj* + *nhead*', etc. In general, hypotheses containing sequences of constituents which can be collected into larger constituents by existing rules are disregarded as redundant (See Figure 4).

[4] **Distinction of lexical categories from other cateogries:** While the general form of CFG

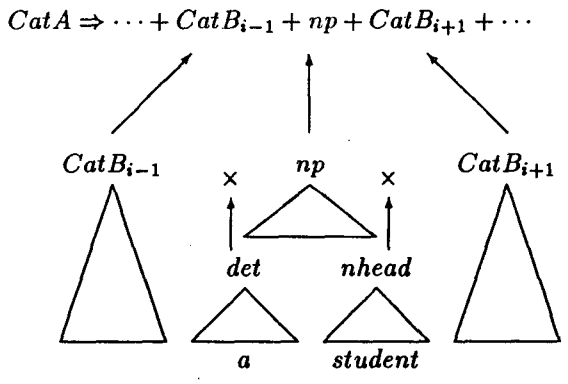


Figure 4: Adjacent Maximal Category

does not distinguish lexical categories from other non-terminals, our grammar does. Therefore, we prohibit GRHP to hypothesize a new rule whose mother category is one of the lexical categories. The lexical categories are allowed only to appear in new lexical rules.

[5] **Distinction of closed and open lexical categories:** We assume that the existing grammar has a complete list of function words. This means that LHSs of rules for new lexical entries are restricted to the open lexical categories, such as noun, verb, adjective, and adverb.

[6] **Use of subcategorization frames:** As in our grammar formalism a subcategorization frame is embedded in the feature structure of a head category, the correspondence between the head category and its subcategories does not appear explicitly in rules. Therefore, a subcategorization frame checking mechanism should be incorporated into the search algorithm and executed before hypothesizing any rule or any lexical entry in order to filter out redundant hypotheses.

[7] **Prohibition of unary rules:** While the general form of CFG allows unary rules and they are sometimes used as category conversion rules in actual descriptions of a grammar, they differ from the constituent rules which specify mother-daughter relationships. For example, a rule '$np \Rightarrow infinitive$' means that an infinitival clause behaves as a noun phrase in larger constituents without changing its structure. Unrestricted introduction of such unary rules, however, increases drastically not only parsing ambiguities but also possible hypotheses generated by GRHP. Except for lexical rules which are unary in nature, we can prohibit unary hypotheses by assuming that the existing grammar exhausts all possible category conversion rules among the categories it uses (See Section 5).

[8] **Distinction of closed and open categories:** We can extend the distinction of open and closed lexical categories in [5] to the other categories. Depending on the completeness of the existing grammar, we can specify a set of categories as closed categories and prohibit GRHP to generate new rules whose RHSs belong to the set.

[9] **Restricted patterns of new rules:** This restriction could be realized by introducing meta-rules which specify the form of a new rule and the relations between adjacent categories. For example, according to the X-bar theory, we can confine a category appearing at the complement position to be a maximal projection.

[10] **Restriction on Lexical Rules:** As we discussed in [7], unary rules are one of the major causes of explosion of the search space. Unary lexical rules can also be restricted by introducing a *prior* knowledge of possible lexical category conversions. For example, while the conversion between a noun and a verb is very frequent in English, the conversion of an adverb with the suffix **-ly** to a verb is extremely rare. This means that, though verb is an open lexical category, we can prohibit a lexical rule which forces a word registered in the dictionary as an adverb to be interpreted as a verb.

5 Preliminary Experiment

To see what sort of hypotheses are actually generated, and how many of them are reasonable (in other words, how many of them are nonsensical), we have conducted a preliminary experiment with the following six sentences.

(1) The girl in the garden has a bouquet.

(2) Buy a new car.

(3) Dogs do dream.

(4) The box is so heavy that I could not move it.

(5) The student has a BMW.

(6) The boy caught several fish.

We deliberately introduce *defects* into the existing grammar which are relevant to the analysis of these sentences. That is, the following rules are removed from the existing grammar for the sake of the experiment.

- pp-attachment rule for noun phrases.
- rule for imperative sentences.
- DO-emphasis rule.
- rule for SO-THAT construction.
- lexical rule for "BMW".
- lexical description for the plural usage of "fish".

The criteria [1]-[5] of redundant hypotheses are included in the basic algorithm of GRHP so that the following lists of hypotheses for these examples do

not contain those which are rejected by these criteria. The hypotheses marked with '→' are the plausible hypotheses. The hypotheses marked by × and ⊗ are the hypotheses removed by adding [6] and [7] as further criteria of redundant hypotheses, respectively. We do not use the criteria of [8]–[10] in this experiment, partly because these are highly dependent on the completeness of the existing grammar and, though very effective for reducing the number of hypotheses, can be arbitrary.

(1) "The girl in the garden has a bouquet."

```
⊗ Rule:  colonp => pp
→ Rule:  np => np,pp
  Rule:  s => np,pp,vp
  Rule:  vp => pp,vp
  Lexical Entry:  v => [in]
```

Instead of the removed pp-attachment rule, '*nhead* ⇒ *nhead* + *pp*', GRHP generates a new pp-attachment rule, '*np* ⇒ *np* + *pp*'.

(2) "Buy a new car."

```
→ ⊗ Rule:  s => vp
```

GRHP generates only one hypothesis, a rule for imperative sentences. This rule looks plausible but the fact that the criteria [7] of redundant hypotheses suppresses this rule indicates that a rule for imperative sentences should not be treated as a normal unary (category conversion) rule but rather a whole-sentencial constituent rule.

(3) "Dogs do dream."

```
× Rule:  ajp => nhead
× Rule:  ajp => vp
⊗ Rule:  colonp => auxdo
⊗ Rule:  colonp => vp
× Rule:  infinitive => nhead
× Rule:  infinitive => vp
  Rule:  np => np,auxdo
  Rule:  np => np,vp
⊗ Rule:  np => relc
⊗ Rule:  np => s
⊗ Rule:  np => vp
  Rule:  s => np,auxdo,nhead
  Rule:  s => np,auxdo,vp
  Rule:  s => np,vp,nhead
  Rule:  s => np,vp,vp
  Rule:  s => relc,nhead
  Rule:  s => relc,vp
  Rule:  s => s,nhead
  Rule:  s => s,vp
⊗ Rule:  sub_clause => nhead
⊗ Rule:  sub_clause => vp
× Rule:  that_clause => nhead
× Rule:  that_clause => vp
  Rule:  vp => auxdo,nhead
→ Rule:  vp => auxdo,vp
⊗ Rule:  vp => auxdo
× Rule:  vppsv => nhead
× Rule:  vppsv => vp
  Lexical Entry:  adj => [dream]
  Lexical Entry:  adv => [dream]
  F Disagrmnt:  np => nhead
  F Disagrmnt:  vp => vp,vp
  F Disagrmnt:  vppsv => v
```

Although this sentence is short, quite a few hypotheses are generated. This is partly because both "do" and "dream" are ambiguous in their parts of speech. Some of the generated hypotheses are based on the interpretation of "dream" as a noun. However, even in the cases in which the main verb is not ambiguous, GRHP always hypothesizes '*vp* ⇒ *vp* + *vp*' as well as the *correct* DO-emphasis rule, as "do" has two parts of speech. As we discuss in the following section, it is impossible to choose one of these hypotheses on the basis of single parsing failures. We need corpus-based techniques to rate the plausibility of these two hypotheses.

(4) "The box is so heavy that I could not move it."

```
× Rule:  ajp => relc,np
× Rule:  ajp => relc
× Rule:  ajp => that_clause
× Rule:  infinitive => ajp,relc,np
× Rule:  infinitive => ajp,relc
× Rule:  infinitive => ajp,that_clause
× Rule:  infinitive => ajp
× Rule:  infinitive => relc,np
× Rule:  infinitive => relc
× Rule:  infinitive => that_clause
  Rule:  nhead => ajp,relc,np
  Rule:  nhead => ajp,relc
  Rule:  nhead => ajp,that_clause
  Rule:  nhead => relc,np
⊗ Rule:  nhead => relc
⊗ Rule:  nhead => that_clause
  Rule:  np => ajp,relc,np
  Rule:  np => ajp,relc
  Rule:  np => ajp,that_clause
  Rule:  s => np,vp,ajp,that_clause
  Rule:  s => np,vp,relc,np
  Rule:  s => np,vp,that_clause
  Rule:  s => s,ajp,relc,np
  Rule:  s => s,ajp,that_clause
  Rule:  s => s,relc,np
→ Rule:  s => s,that_clause
  Rule:  sub_clause => ajp,relc,np
  Rule:  sub_clause => ajp,that_clause
  Rule:  sub_clause => relc,np
⊗ Rule:  sub_clause => that_clause
× Rule:  that_clause => ajp,relc,np
× Rule:  that_clause => ajp,relc
× Rule:  that_clause => ajp,that_clause
× Rule:  that_clause => ajp
× Rule:  vp => adv,ajp,relc,np
× Rule:  vp => adv,ajp,relc
```

```
× Rule:  vp => adv,ajp,that_clause
× Rule:  vp => adv,ajp
× Rule:  vp => ajp,relc,np
× Rule:  vp => ajp,relc
× Rule:  vp => ajp,that_clause
× Rule:  vp => ajp
× Rule:  vp => relc,np
× Rule:  vp => relc
× Rule:  vp => that_clause
× Rule:  vp => vp,relc,np
× Rule:  vp => vp,relc
× Rule:  vppsv => adv,ajp,relc,np
× Rule:  vppsv => adv,ajp,relc
× Rule:  vppsv => adv,ajp,that_clause
× Rule:  vppsv => adv,ajp
× Rule:  vppsv => ajp,relc,np
× Rule:  vppsv => ajp,relc
× Rule:  vppsv => ajp,that_clause
× Rule:  vppsv => ajp
× Rule:  vppsv => relc,np
× Rule:  vppsv => relc
× Rule:  vppsv => that_clause
  Lexical Entry:  adj => [that]
  Lexical Entry:  adv => [heavy]
  Lexical Entry:  adv => [that]
  Lexical Entry:  n => [heavy]
  Lexical Entry:  n => [so]
  Lexical Entry:  n => [that]
  Lexical Entry:  v => [heavy]
  Lexical Entry:  v => [so]
  Lexical Entry:  v => [that]
  F Disagrmnt:  ajp => ajp,that_clause
  F Disagrmnt:  sub_clause => conj3,s
  F Disagrmnt:  vp => vp,ajp
  F Disagrmnt:  vp => vp,np
→ F Disagrmnt:  vp => vp,that_clause
```

In this example, '*vp* ⇒ *vp* + *that_clause*' (or '*s* ⇒ *s* + *that_clause*') could be the appropriate hypothesis. However, simple addition of such a rule to the existing grammar results in overgeneralization. The rule should have a condition on the existence of "so" in '*vp*' (or '*s*') while a similar effect can also be attained by adding a new lexical entry for "heavy" which has a subcategorization frame containing a 'that clause'. That is, the system has to decide which hypothesis is more plausible, either "heavy" can subcategorize a 'that clause' or "so" is crucial in making '*vp*' to be related with a 'that clause'. This decision may not be possible, if this sentence is the only one sentence in a corpus which contains this construction. Like Example 3, we need corpus-based techniques to choose the right one.

(5) "The student has a BMW."

```
→ Lexical Entry:  n => ['BMW']
```

GRHP generates the correct hypothesis which assigns the expected lexical category to the unregistered word.

Sample Sentence	Number of Hypotheses			
	NR	LE	FD	Total
(1)	4	1	0	5
(2)	1	0	0	1
(3)	28	2	3	33
(4)	58	9	5	72
(5)	0	1	0	1
(6)	8	2	1	11

NR: New Rule
LE: New Lexical Entry
FD: Feature Disagreement

Table 2: Number of Hypotheses

(6) "The boy caught several fish."

```
× Rule:  ajp => det,nhead
× Rule:  ajp => det
× Rule:  infinitive => det,nhead
  Rule:  s => np,vp,det,nhead
  Rule:  s => relc,det,nhead
× Rule:  that_clause => det,nhead
× Rule:  vp => det,nhead
× Rule:  vppsv => det,nhead
  Lexical Entry:  adj => [several]
  Lexical Entry:  n => [several]
→ F Disagrmnt:  np => det,nhead
```

GRHP generates the correct hypothesis of the feature disagreement between the plural determiner "several" and the noun "fish" as one of possible hypotheses.

Table 2 summarizes the number of hypotheses generated for each sample sentence. As can be seen, while appropriate hypotheses are generated, quite a few other hypotheses are also generated, especially in the case of the third and the fourth sentences.

However, as shown in Table 3, the criteria [6] and [7] of redundant hypotheses can eliminate significant portions of nonsensical hypotheses (Table 3 shows the effects of these criteria on the number of hypothesized new rules). In Example (4), for example, 31 out of 58 initially hypothesized rules are eliminated by [6] and [7], while 16 out of 28 rules are eliminated in Example (3). Furthermore, we expect that introduction of other criteria for redundant elimination based on [8]–[10] will reduce the number of hypotheses significantly and make the succeeding stage of the corpus-based statistical analysis feasible.

The experiment on another set of sample sentences from the UNIX on-line manual confirms our expectation (See Table 4). The number of hypotheses generated in this experiment is very much similar to that of the experiment on artificial samples (note that Table 4 shows the number of hypotheses generated before elimination by the criteria [6] and [7]).

Sample Sentence	Number of New Rules		
	[1]-[5]	[1]-[6]	[1]-[7]
(1)	4	4	3
(2)	1	1	0
(3)	28	20	12
(4)	58	20	17
(5)	0	0	0
(6)	8	2	2

Table 3: Effects of Redundancy Elimination

6 Corpus-based Techniques and Linguistic Knowledge Acquisition

We discussed that using an existing grammar should enable us to avoid a huge search space which grammatical learning would otherwise have. Instead of inducing grammatical concepts from scratch, our framework uses the categories prepared in an existing grammar for formulating new structural rules.

However, ***linguistic knowledge acquisition*** is inherently an inductive process. We cannot expect GRHP alone to choose correct hypotheses without observing analysis results of other sentences in a corpus.

Although we have not yet implemented the corpus-based component, the result of the preliminary experiment indicates what sorts of functions this component should have.

[1] In Example (6), we have a feature disagreement hypothesis for "several fish" and two lexical hypotheses for "several". Further analysis of the feature disagreement hypothesis will lead to two competing hypotheses, one of which requires a revised lexical description of "several" and the other of which suggests that of "fish". The other two lexical hypotheses also suggest different revisions in the description of "several". However, the analysis of this sentence alone may not enable us to decide which of these four hypotheses is the right one.

We reported in [Tsujii *et al.*, 1992] that a simple statistical measure like the *Failure Rate of a Word* (ratio of the number of sentences containing a word that cannot be parsed to the total number of sentences containing the same word) is useful for discovering words whose lexical descriptions contain defects. This kind of simple measures would also be effective in a situation like Example (6). That is, we can expect that, while the frequency of the word "several" would be high, the frequency of the hypotheses suggesting the revisions of the lexical descriptions of this word would be relatively low.

[2] As we noted in the comment on Example (3), whenever DO-emphasis construction appears, the same pair of the hypotheses, '$vp \Rightarrow vp + vp$' and '$vp \Rightarrow auxdo + vp$', will be generated. Unless other types of failures lead to one of these hypotheses, they would be judged to have exactly the same remedial powers, i.e. the same set of failures are restored by them. In such a situation, we may be able to choose the right one by comparing the specificities of competing hypotheses. In this example, the former hypothesis which uses '*vp*' instead of '*auxdo*' can be judged as having excessive generative powers and therefore inappropriate because the other competing hypothesis with far restricted generative powers can restore the same set of parsing failures.

In order for such comparison to be meaningful, the system first have to judge, by corpus-based techniques, whether competing hypotheses have the same remedial powers or not. If the more general ones appear frequently as remedial rules for parsing failures which cannot be restored by the specific ones, the general ones would be the right ones.

[3] Example (4) shows a situation opposite to Example (3). We have two (or three) viable competing hypotheses in this example. One is the specific hypothesis with very restricted generative powers which suggests to revise the lexical description of "heavy". The other is a more general hypothesis which allows '*vp*' (or '*s*') to be followed by '*that_clause*'. Although either of these two can restore the parsing failure of this sentence, the specific one cannot restore parsing failures in other sentences in which SO-THAT constructions appear with different adjectives. That is, unlike Example (3), these two hypotheses have different remedial powers and, because of this, the general one should be chosen as the right one.

Furthermore, though simple addition of this general rule results in serious over-generalization, to curb this over-generalization needs complex revisions of related grammar rules in order for a feature indicating the existence of "so" to be percolated to the node of '*vp*' (or '*s*'). Such invention of a new feature and re-organization of related rules seem beyond the current framework and we expect human linguists to examine suggested hyoptheses.

7 Conclusion

We proposed in this paper a new framework which acquires linguistic knowledge from parsing failures. Linguistic knowledge acquisition been studied so far by two extreme approaches. One approach assumes very little prior knowledge and tries to induce most of linguistic knowledge from scratch, while the other assumes existence of almost complete knowledge and tries only to learn the probabilistic properties from corpora. Our approach is between these two extremes. Although it assumes existence of rather comprehensive linguistic knowledge, it tries to create new units of knowledge which deal with specificities of given sublanguages.

Considering the diverse nature of *sublanguages* and the essential difficulties involved in inductive processes, we believe that our approach has practical advantages over the other approaches as well as interesting theoretical implications. However, the re-

Sample Sentence	Number of Hypothese s			
	NR	LE	FD	Total
Variables are initialized to the null string.	12	8	3	23
The default blocking factor is 20 blocks.	27	3	1	31
There is no way selectively to follow symbolic links.	19	6	1	26
When closed, clock displays a clock face.	1	0	0	1
The default is DELETE.	0	4	0	4
This support is normally invisible to the user.	26	13	3	42
The output device in use is not capable of backspacing.	40	14	3	57
As a result, the first line must not have any superscripts.	13	3	0	16
Pathnames are restricted to 128 characters.	0	1	0	1
They default to the standard input and the standard output.	12	5	1	18
Remove initial definitions for all predefined symbols.	10	2	0	12
Remove any definition for the symbol name.	2	0	0	2
The most recent command is retained in any case.	82	11	5	98
Such loops are detected, and cause an error message.	13	0	0	13
Components of an expression are separated by white space.	2	0	0	2
The kernel then attempts to overlay the new process with the desired program.	8	5	0	13

Table 4: Number of Hypotheses (Sentences from the UNIX manual)

search of this direction has just started and quite a few problems remain to be solved. The following shows some of these problems.

- **Analysis Methods of Feature Disagreements:** Unlike robust parsing of ill-formed input, we have to identify real causes of disagreements and create a set of sub-hypotheses on real causes. In many cases, feature disagreements are caused by lack of or improper lexical descriptions.
- **Plausibility Rating of Hypotheses:** As we saw in Section 6, the corpus-based component has to take into consideration several factors, such as remedial powers and specificities of individual hypotheses, relative frequencies of hypotheses (like fault rates), competing relationships among them, etc. in order to rate the plausibility of individual hypotheses. However, the observation in Section 6 is still very sketchy. In order to design the corpus-based component, we need more detailed observation of the nature of hypotheses generated by GRHP.
- **Further Restrictions on Viable Hypotheses:** Although the current criteria of redundant hypotheses reduce significantly the number of hypotheses, there still remain cases where more than thirty hypotheses are generated.
- **Refinement of Generated Hypotheses:** The current version of GRHP only generates structural skeletons of new rules. These structural skeletons should be accompanied by conditions on features. In particular, it would be crucial in practical applications for GRHP to generate hypotheses of lexical descriptions with fuller feature specifications.

Acknowledgements

We would like to thank our colleagues at CCL who are interested in corpus-based techniques. Their comments on the paper were very useful. We would also thank Mr. Tomoki Tsumura, Dr. Katsura Kawakami and the colleagues at Matsushita, who allowed Kiyono to do research at CCL.

References

[Ananiadou, 1990] Sofia Ananiadou. Sublanguage studies as the basis for computer support for multilingual communication. In *Proc. of Termplan '90*, Kuala Lumpur, 1990.

[Douglas and Dale, 1992] Shona Douglas and Robert Dale. Towards robust patr. In *Proc. of COLING-92*, pages 468–474, 1992.

[Goeser, 1992] Sebastian Goeser. Chart parsing of robust grammars. In *Proc. of COLING-92*, pages 120–126, 1992.

[Mellish, 1989] Chris S. Mellish. Some chart-based techniques for parsing ill-formed input. In *Proc. of the 27th ACL meeting*, pages 102–109, 1989.

[Sekine *et al.*, 1992] Satoshi Sekine, et al. Linguistic knowledge generator. In *Proc. of COLING-92*, pages 560–566, 1992.

[Strzalkowski, 1992] Tomek Strzalkowski. Ttp: A fast and robust parser for natural language. In *Proc. of COLING-92*, pages 198–204, 1992.

[Tsujii *et al.*, 1992] Jun-ichi Tsujii, et al. Linguistic knowledge acquisition from corpora. In *Proc. of 2nd FGNLP*, pages 61–81, UMIST, 1992.