Effective Analysis of Causes and Inter-dependencies of Parsing Errors

Tadayoshi Hara[1] **Yusuke Miyao**[1] **Jun'ichi Tsujii**[1,2,3]
[1]Department of Computer Science, University of Tokyo
Hongo 7-3-1, Bunkyo-ku, Tokyo, 113-0033, JAPAN
[2]School of Computer Science, University of Manchester
[3]NaCTeM (National Center for Text Mining)
{harasan,yusuke,tsujii}@is.s.u-tokyo.ac.jp

Abstract

In this paper, we propose two methods for analyzing errors in parsing. One is to classify errors into categories which grammar developers can easily associate with defects in grammar or a parsing model and thus its improvement. The other is to discover inter-dependencies among errors, and thus grammar developers can focus on errors which are crucial for improving the performance of a parsing model.

The first method uses patterns of errors to associate them with categories of causes for those errors, such as errors in scope determination of coordination, PP-attachment, identification of antecedent of relative clauses, etc. On the other hand, the second method, which is based on re-parsing with one of observed errors corrected, assesses inter-dependencies among errors by examining which other errors were to be corrected as a result if a specific error was corrected.

Experiments show that these two methods are complementary and by being combined, they can provide useful clues as to how to improve a given grammar.

1 Introduction

In any kind of complex systems, analyzing causes of errors is a crucial step for improving its performance. In recent sophisticated parsing technologies, the step of error analysis has been becoming more and more convoluted and time-consuming, if not impossible. While common performance evaluation measures such as F-values are useful to compare the performance of systems or evaluate improvement of a system, they hardly give useful clues as to how to improve a system. Evaluation measures usually assume uniform units such as the number of correctly or incorrectly recognized constituent boundaries and their labels, or in a similar vein, dependency links among words and their labels, and then compute single values such as the F-value. These values do not give any insights as to where the weaknesses exist in a parsing model. As a result, the improvement process takes the form of time consuming trial-error cycles.

Once grammar developers know the actual distribution of errors across different categories such as PP-attachment, complement/adjunct distinction, gerund/participle distinction, etc., they can think of focused and systematic improvement of a parsing model.

Another problem of the F-value in terms of uniform units is that it does not take inter-dependencies among errors into consideration. In particular, for parsers based on grammar formalisms such as LFG (Kaplan and Bresnan, 1995), HPSG (Pollard and Sag, 1994), or CCG (Steedman, 2000), units (eg. single predicate-argument links) are inter-related through hierarchical structures and structure sharing assumed by these formalisms. Single errors are inherently propagated to other sets of errors. This is also the case, though to a lesser extent, for parsing models in which shallow parsing is followed by another component for semantic label assignment.

In order to address these two issues, we propose two methods in this paper. One is to recognize cause categories of errors and the other is to capture inter-dependencies among errors. The former method defines various patterns of errors to identify categories of error causes. The latter method re-parses a sentence with a single target error corrected, and regards the errors which are corrected in re-parse as errors dependent on the target.

Although these two methods are implemented for a specific parser using HPSG (Miyao and Tsujii, 2005; Ninomiya et al., 2006), the same ideas can be applied to any type of parsing models.

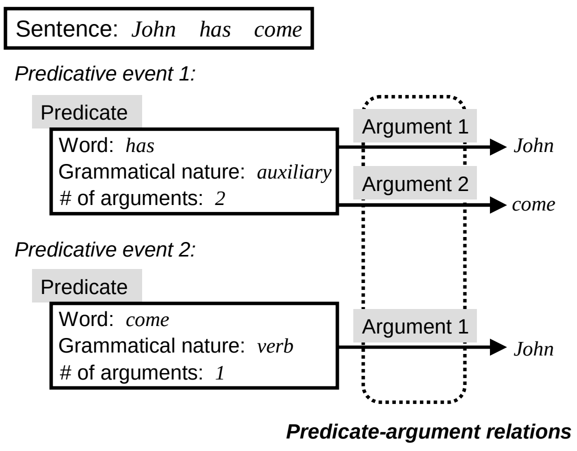


Figure 1: Predicate-argument relations

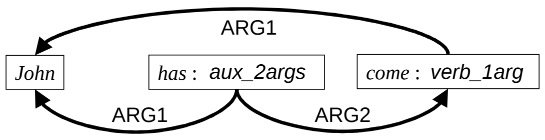


Figure 2: Representation of predicate-argument relations

In the following, Section 2 introduces a parser and its evaluation metrics, Section 3 illustrates difficulties in analyzing parsing errors based on common evaluation measures, and Section 4 proposes the two methods for effective error analysis. Section 5 presents experimental results which show how our methods work for analyzing actual parsing errors. Section 6 and Section 7 illustrate further application of these methods to related topics. Section 8 summarizes this research and indicates some of future directions.

2 A parser and its evaluation

A parser is a system which interprets given sentences in terms of structures derived from syntactic or in some cases semantic viewpoints, and structures constructed as a result are used as essential information for various tasks of natural language processing such as information extraction, machine translation, and so on.

In this paper, we address issues involved in improving the performance of a parser which produces structural representations deeper than surface constituent structures. Such a parser is called a "deep parser." In many deep parsers, the output structure is defined by a linguistics-based grammar framework such as CFG, CCG (Steedman, 2000), LFG (Kaplan and Bresnan, 1995) or HPSG (Pollard and Sag, 1994). Alternatively, some deep parsing models assume staged processing in which a stage of shallow parsing is followed by a stage of semantic role labeling, which assigns labels indicating semantic relationships between predicates and their arguments. In either case, we assume a parser to produce a single "deep" structural representation for a given sentence, which is chosen from a set of possible interpretations as the most probable one by a disambiguation model.

Abbr.	Full	Abbr.	Full
aux	auxiliary	*conj*	conjunction
prep	prepositional	*lgs*	logical subject
verb	verb	*app*	apposition
coord	coordination	*relative*	relative
det	determiner	*_Narg(s)*	takes N arguments
adj	adjunction	*_mod*	modifies a word

Table 1: Descriptions for predicate types

For evaluation of the performance of a parser, various metrics have been introduced according to the structure captured by a given grammar formalism or a system of semantic labels. In most cases, instead of examining correctness for a whole structure, a parser is evaluated in terms of the F-value which shows how correctly it recognizes relationships among words and assigns "labels" to the relationships in the structure. In this paper, we assume a certain type of "predicate-argument relation."

In this measurement, a structure given for a sentence is decomposed into a set of predicative words and their arguments. A predicate takes other words as its arguments. In our representation, the arguments are labeled by semantically neutral labels such as ARG$n(n = 1...5)$ and MOD. In this representation, a basic unit is a triplet, such as

<Predicate:PredicateType,
ArgumentLabel,
Argument>,

where "Predicate" and "Argument" are surface words. As shown in the examples in Section 4, "PredicateType" bears extra information concerning the syntactic construction in which the triplet is embedded. ARG1-ARG5 express relations between a Head and its complement, while MOD expresses a relation between an Adjunct and its modifiee. Since all dependency relations are expressed by triplets, triplets contain not only semantic de-

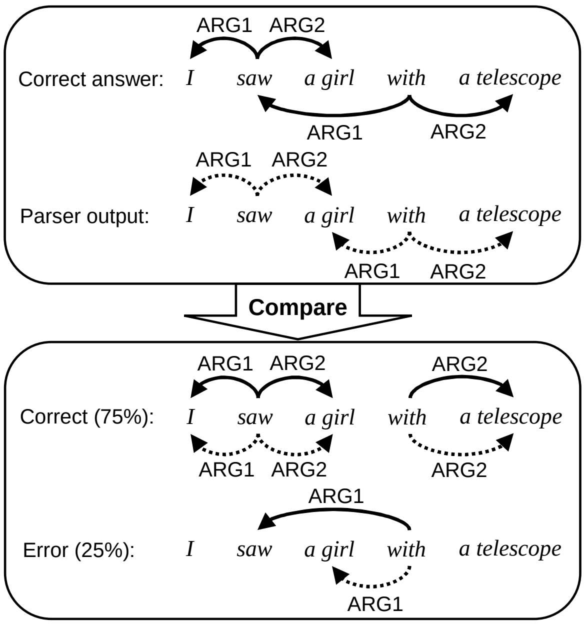


Figure 3: An example of parsing performance evaluations

pendencies but also many dependencies which are essentially syntactic in nature. Figure 1 shows an example used in Miyao and Tsujii (2005) and Ninomiya et al. (2006).

This example shows predicate-argument relations for "*John has come.*" There are two predicates in this sentence, "*has*" and "*come*". The word "*has*", which is used as an auxiliary verb, takes two words, "*John*" and "*come*", as its arguments, and therefore two triplets of predicate-argument relation, <*has* ARG1 *John*> and <*has* ARG2 *come*>. As for the predicative word "*come*", we have one triplet <*come* ARG1 *John*>. Note that, in this HPSG analysis, the auxiliary verb "*has*" is analyzed in such a way that it takes one NP as subject and one VP as complement, and that the subject of the auxiliary verb is shared by the verb ("*come*") in VP as its subject (Figure 2). The fact that "*has*" in this sentence is an auxiliary verb is indicated by the "PredicateType", *aux_2args*. A "PredicateType" consists of a type and the number of arguments it takes (Table 1).

3 Difficulties in analyzing parsing errors

Figure 3 shows an example of the evaluation of the parser based on these predicate-argument relations. Note that the predicate types are abbreviated in this figure. In the sentence "*I saw a girl with a telescope*", there should be four triplets for the two predicates, "*saw*" and "*with*," each of which takes

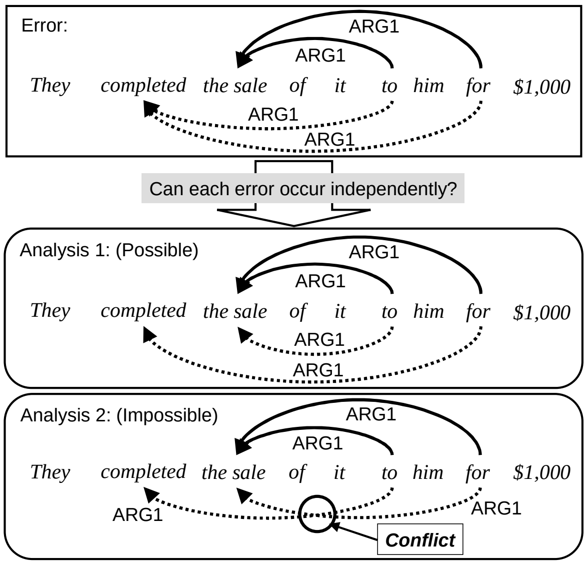


Figure 4: Sketch of error propagation

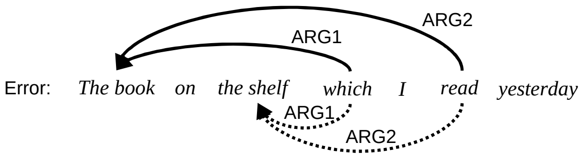


Figure 5: Parsing errors around one relative clause attachment

two arguments. Although the parser output does indeed contain four triplets, the first argument of "*with*" is not the correct one. Thus, this output is erroneous, with the F-value of 75%.

While the F-value thus computed is fine for capturing the performance of a parser, it does not offer any help for improving its performance.

First, because it does not give any indication on what portion of erroneous triplets are in PP-attachment, complement/adjunct distinction, gerund/participle distinction, etc., one cannot determine which part of a parsing model should be improved. In order to identify error categories, we have to manually compare a parsing output with a correct parse and classify them. Consider again the example in Figure 3. We can easily observe that "ARG1" of predicate "*with*" was mistaken. In this case, the word linked via "ARG1" represents a modifiee of the prepositional phrase, and thereby we conclude that the error is in PP-attachment. While the process looks straightforward for this simple sentence and error, to perform such a manual inspection for all sentences and more complex types of errors is costly, and becomes inhibitive when the size of a test set of sentences is realistically large.

Another problem with the F-value is that it ignores inter-dependencies among errors. Since the F-value does not consider inter-dependencies, one cannot determine which errors are more crucial than others in terms of the performance of the system as a whole.

A simple example of inter-dependency is shown in Figure 4. "ARG1" of "*for*" and "*to*" were mistaken by a parser, both of which can be classified as PP-attachments as in Figure 3. However, the two errors are not independent. The former error can occur by itself (Analysis 1) while the latter cannot because of the structural conflict with the former (Analysis 2). The occurrence of the latter error thus forces the former.

Moreover, inter-dependency in a deep parser based on linguistics-based formalisms can be complicated. Error propagation is ingrained in grammar itself. Consider Figure 5. In this example, a wrong decision on the antecedent of a relative clause results in a wrong triplet of the predicate in the embedded clause with the antecedent. That is, the two erroneous triplets, one of the "ARG1" of "*which*" and the other of the "ARG2" of "*read*," were caused by a single wrong decision of the antecedent of a relative clause. Such a propagation of errors can be even more complicated, for example, when the predicate in the relative clause is a control verb.

In the following section we propose two methods for analyzing errors. Although both methods are implemented for the specific parser *Enju* (Miyao and Tsujii, 2005; Ninomiya et al., 2006), the same ideas can be implemented for any parsing model.

4 Methods for effective error analysis

4.1 Recognizing categories of error causes

While the Enju parser produces rich feature structures as output, the performance is evaluated by the F-value in terms of basic units of predicate-argment structure. As we illustrated in Section 2, the basic unit is a triplet in the following form.

<Predicate:PredicateType,
  ArgumentLabel,
  Argument>

We illustrated in Section 2 how we can identify errors in PP-attachment simply by examining a triplet produced by the parser with the corresponding triplet in the gold standard parse.

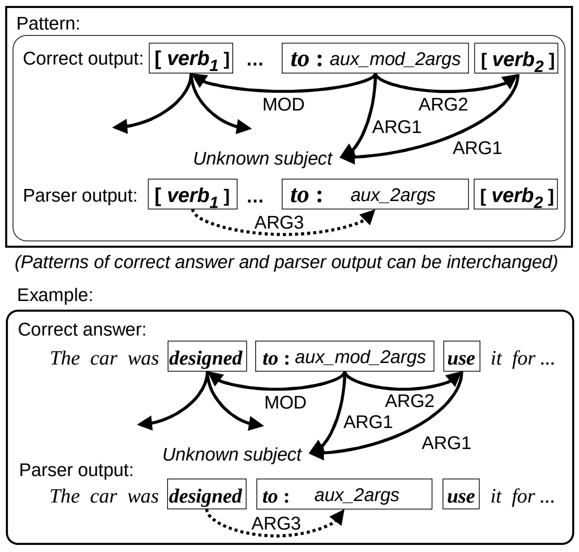

Pattern:
Correct output: [verb1] ... to : aux_mod_2args [verb2]
MOD
ARG2
ARG1
ARG1
Unknown subject
Parser output: [verb1] ... to : aux_2args [verb2]
ARG3
(Patterns of correct answer and parser output can be interchanged)
Example:
Correct answer:
The car was designed to : aux_mod_2args use it for ...
MOD
ARG2
ARG1
ARG1
Unknown subject
Parser output:
The car was designed to : aux_2args use it for ...
ARG3

Figure 6: Pattern for "To-infinitive for modifier/argument of verb"

However, in more complex cases, we have to consider a set of mismatched triplets collectively in order to map errors to meaningful error causes. The following are typical examples of error causes and pattern rules which identify them.

(1) Interpretation of Infinitival Clauses as Adjunct or Complement

Two different types of interpretations of the infinitival clauses are explicitly indicated by "PredicateType." Consider the following two sentences.

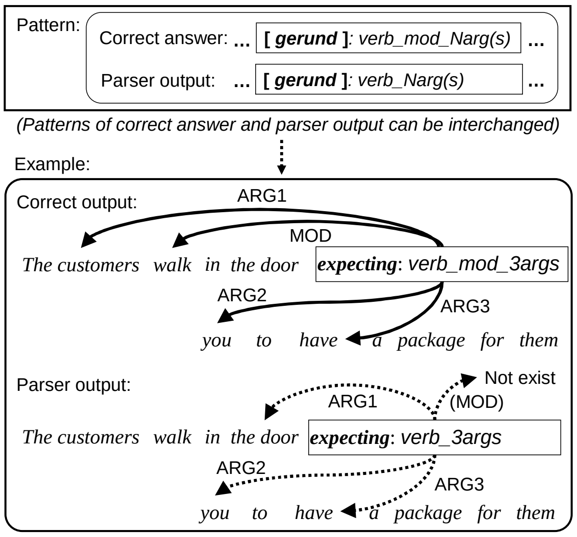
(a) [Infinitival clause as an adjunct of the main clause]

The car was designed (by John) to use it for business trips.

(b) [Infinitival clause as an argument of catenative verb]

The car is designed to run fast.

In both sentences, "*to*" is treated as a predicate to represent the infinitival clauses in triplets. However, Enju marks the "PredicateType" of (a) as "*aux-mod-2args*," while it marks the predicate simply as "*aux-2args*" in (b). Furthermore, the linkage between the main clause and the infinitival clause is treated differently. In (a), the infinitival clause takes the main clause with relation MOD, while in (b) the main clause takes the infinitival



Pattern:
Correct answer: ... [gerund]: verb_mod_Narg(s) ...
Parser output: ... [gerund]: verb_Narg(s) ...
(Patterns of correct answer and parser output can be interchanged)
Example:
Correct output:
ARG1
MOD
The customers walk in the door expecting: verb_mod_3args
ARG2
ARG3
you to have a package for them
Parser output:
Not exist (MOD)
ARG1
The customers walk in the door expecting: verb_3args
ARG2
ARG3
you to have a package for them

Figure 7: Pattern for “Gerund acts as modifier or not”

clause as ARG3. Furthermore, in the catenative verb interpretation of “*designed*”, the deep object (the surface subject in this example) fills ARG1 of the verb in the infinitival clause (complement), while in the adjunct interpretation, the deep subject which is missing in this sentence occupies the same role. Consequently, a single erroneous choice between these two interpretations results in a set of mismatched triplets.

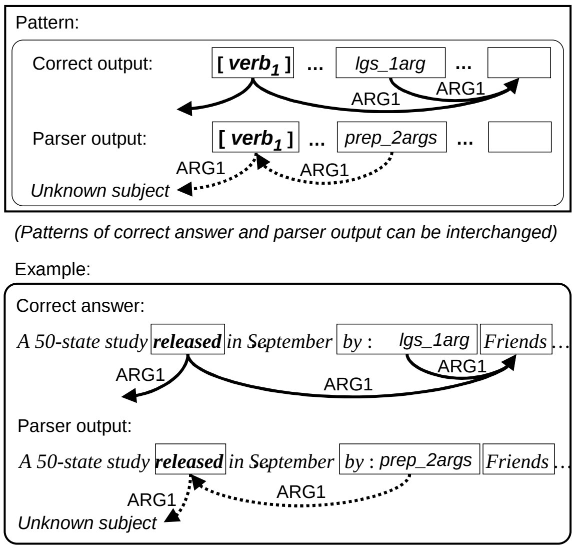
We recognize such a set of mismatched triplets by a pattern rule (Figure 6) and map them to this type of error cause.

(2) Interpretation of Gerund-Participle interpretations

A treatment similar to (1) is taken for different interpretations of Gerund. Interpretation as Adjunct of a main clause is signaled by the “PredicateType” *verb-mod-**, while an interpretation as a modifier of a noun is represented by the “PredicateType” *verb* (Figure 7).

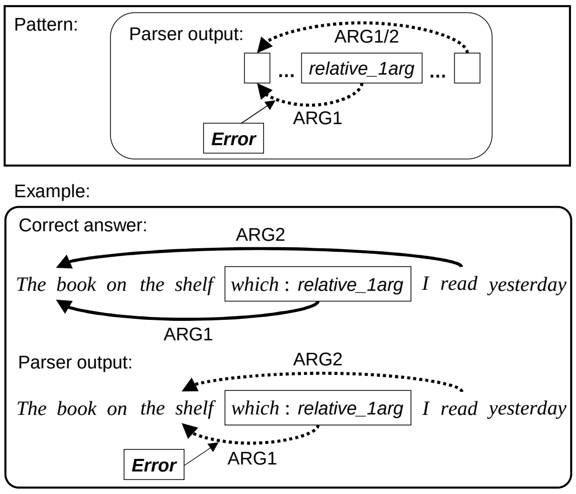
(3) Interpretation of “*by*”

A prepositional phrase with “*by*” in a passive clause can be interpreted as a deep subject, while the same phrase can be interpreted as an ordinary PP phrase that is used as an adjunct. The first interpretation is marked by the “PredicateType” *lgs* (logical subject) which takes only one argument. The relationship between the passivized verb and the deep subject is captured by ARG1 which goes from the verb to the noun phrase. On the other hand, in the interpretation as an ordinary PP, the preposition as predicate links the main verb and NP via ARG1 and ARG2, respectively (Figure 8).



Pattern:
Correct output: [verb1] ... lgs_1arg ...
ARG1
ARG1
Parser output: [verb1] ... prep_2args ...
ARG1
ARG1
Unknown subject
(Patterns of correct answer and parser output can be interchanged)
Example:
Correct answer:
A 50-state study released in September by : lgs_1arg Friends ...
ARG1
ARG1
ARG1
Parser output:
A 50-state study released in September by : prep_2args Friends ...
ARG1
ARG1
Unknown subject

Figure 8: Pattern for “Subject for passive sentence or not”



Pattern:
Parser output:
ARG1/2
... relative_1arg ...
ARG1
Error
Example:
Correct answer:
ARG2
The book on the shelf which : relative_1arg I read yesterday
ARG1
Parser output:
ARG2
The book on the shelf which : relative_1arg I read yesterday
Error
ARG1

Figure 9: Pattern for “Relative clause attachment”

Again, a set of mismatched triplets should be mapped to a single cause of errors via a pattern rule.

(4) Antecedent of a Relative Clause

This type of error is manifested by two mismatched triplets with different predicates. This is because a wrong choice of antecedent for a relative clause results in a wrong link for the trace of the relative clause.

Since a relative clause pronoun is treated as a

Cause categories	Patterns
[Argument selection]	
Prepositional attachment	ARG1 of *prep_**
Adjunction attachment	ARG1 of *adj_**
Conjunction attachment	ARG1 of *conj_**
Head selection for noun phrase	ARG1 of *det_**
Coordination	ARG1/2 of *coord_**
[Predicate type selection]	
Preposition/Adjunction	*prep_** ↔ *adj_**
Gerund acts as modifier/not	*verb_mod_Narg(s)* ↔ *verb_Narg(s)*
Coordination/conjunction	*coord_** ↔ *conj_**
# of arguments for preposition	*prep_Marg(s)* ↔ *prep_Narg(s)*
Adjunction/adjunctive noun	*adj_** ↔ *noun_**
[More structural errors]	
To-infinitive for modifier/argument of verb	see Figure 6
Subject for passive sentence/not	see Figure 8
[Others]	
Comma	any error around ","
Relative clause attachment	see Figure 9

Table 2: Defined patterns for cause categories

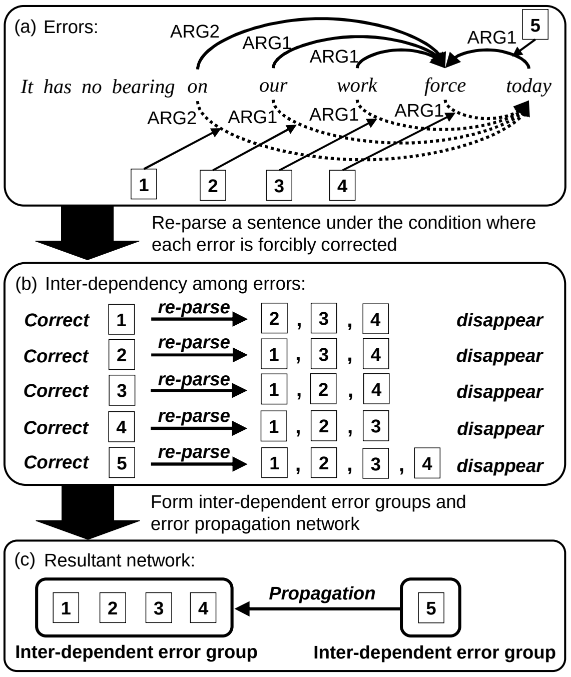

(a) Errors:
ARG2
ARG1
ARG1
ARG1
5
It has no bearing on our work force today
ARG2
ARG1
ARG1
ARG1
1
2
3
4
Re-parse a sentence under the condition where each error is forcibly corrected
(b) Inter-dependency among errors:
Correct 1 re-parse 2, 3, 4 disappear
Correct 2 re-parse 1, 3, 4 disappear
Correct 3 re-parse 1, 2, 4 disappear
Correct 4 re-parse 1, 2, 3 disappear
Correct 5 re-parse 1, 2, 3, 4 disappear
Form inter-dependent error groups and error propagation network
(c) Resultant network:
1 2 3 4
Propagation
5
Inter-dependent error group
Inter-dependent error group

Figure 10: Schema of capturing inter-dependencies

predicate which takes the antecedent as its single argument, identification of error type can be done simply by looking at ARG1. However, since the errors usually propagate to the triplets that contain their traces, we have to map them together to the single error (Figure 9).

Table 2 shows the errors across different types which our current version of pattern rules can identify.

4.2 Capturing inter-dependencies among errors

Some inter-dependencies among erroneous triplets are ingrained in grammar, such as the case of antecedent of a relative clause in (4) in Section 4.1. Some are caused by general constraints such as the projection principle in dependency structure (Figure 4 in Section 2).

Regardless of causes of dependencies, to recognize inter-dependencies among errors is a crucial step of effective error analysis.

Our method consists of the following four steps:

[Step 1] Re-parsing a target sentence: A given sentence is re-parsed under the condition where an error is forcibly corrected.

[Step 2] Forming a network of inter-dependencies of errors: By comparing the new parse result (a set of triplets) with the initial parse result, this step creates a directed graph of errors in the initial parse. A directed link shows that correction of the error in the starting node produces a new parse result in which the error in the receiving node of the link disappears.

[Step 3] Forming groups of inter-dependent errors: This step recognizes a group of inter-dependent errors which forms a directed circle in the network created by [Step 2].

[Step 4] Forming a network of error propagation: This step creates a new network by reducing each of inter-dependent error groups of [Step 3] to a single node.

Figure 10 illustrates how these steps work. In this example, while "*today*" should modify the noun phrase "*our work force*", the initial parse wrongly takes "*today*" as the head noun of the whole noun phrase. As a result, there are five errors; three wrong outputs, "ARG2" of "*on*" (Error 1), "ARG1" of "*our*" (Error 2) and "ARG1" of "*work*" (Error 3). There is an extra triplet for "ARG1" of "*force*" (Error 4), and a triplet for "ARG1" of "today" (Error 5) is missing (Figure 10 (a)).

Figure 10 (b) shows inter-dependencies among the errors recognized by [Step 2], and Figure 10

Cause categories of errors	# of	
	Errors	Locations
Classified	**2,078**	**1,671**
[Argument selection]		
Prepositional attachment	579	579
Adjunction attachment	261	261
Conjunction attachment	43	40
Head selection for noun phrase	30	30
Coordination	202	184
[Predicate type selection]		
Preposition/Adjunction	108	54
Gerund acts as modifier/not	84	31
Coordination/conjunction	54	27
# of arguments for preposition	51	17
Adjunction/adjunctive noun	13	13
[More structural errors]		
To-infinitive for modifier/argument of verb	120	22
Subject for passive sentence/not	8	3
[Others]		
Comma	444	372
Relative clause attachment	102	38
Unclassified	**2,631**	–
Total	**4,709**	–

Table 3: Errors classified into cause categories

(c) shows what the resultant network looks like. An inter-dependent error group of 1, 2, 3 and 4 is recognized by [Step 3] and represented as a single node. Error 5 is propagated to this node in the final network.

5 Experiments

We applied our methods to the analyses of actual errors produced by Enju. This version of Enju was trained on the Penn Treebank (Marcus et al., 1994) Section 2-21.

5.1 Observation of identified cause categories

We first parsed sentences in PTB Section 22, and based on the observation of errors, we defined the patterns in Section 4. We then parsed sentences in Section 0. The errors in Section 0 were mapped to error cause categories by the pattern rules created for Section 22.

Table 3 summarizes the distribution across the causes of errors. The left and right numbers in the table show the number of erroneous triplets classified into the categories and the frequency of the patterns matched, respectively. The table shows that, with the 14 pattern rules, we successfully observed 1,671 hits and 2,078 erroneous triplets are dealt with by these hits. This amounts to more than 40% erroneous triplets (2,078/4,709). Since this was the first attempt, we expect the coverage can be easily improved by adding new patterns.

Evaluated sentences (erroneous)	1,811 (1,009)
Errors (Correctable)	4,709 (3,085)
Inter-dependent error groups	1,978
Correction propagations	501
F-score (LP/LR)	90.69 (90.78/90.59)

Table 4: Summary of inter-dependencies

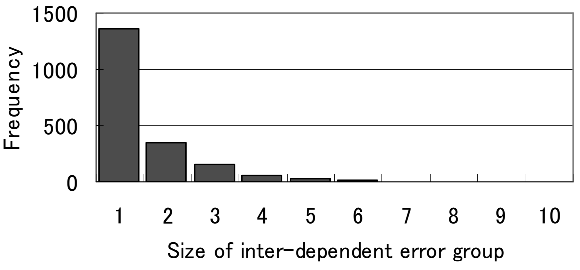


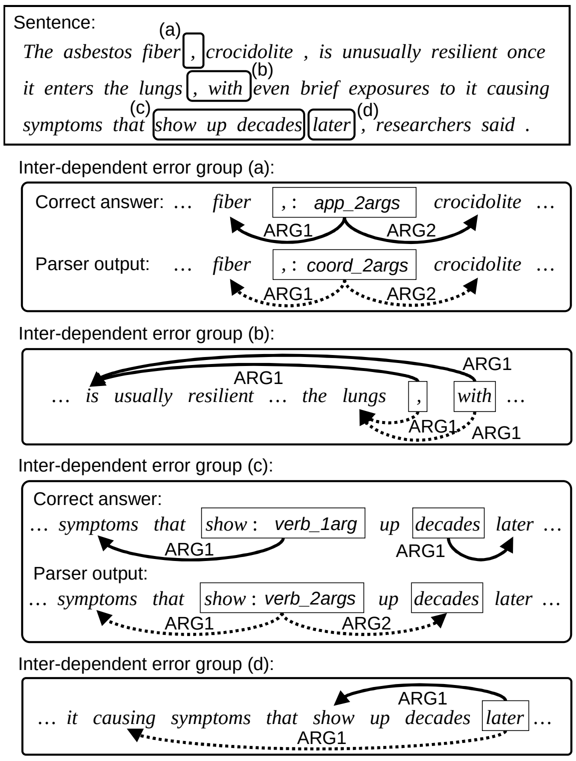
Figure 11: Frequency of each size of inter-dependent error group

From the table, we can observe that a significant portion of errors is covered by simple types of error causes such as PP-attachment and Adjunct attachment. They are simple in the sense that the number of erroneous triplets treated and the frequency of the pattern application coincide. However, their conceived significance may be overrated. These simple types may constitute parts of more complex error causes. Furthermore, since pattern rules for these simple causes are easy to prepare and have already been covered by the current version, most of the remaining 60% of the erroneous triplets are likely to require patterns for more complex causes.

On the other hand, patterns for complex causes collect more erroneous triplets once they are fired. This tendency is more noticeable in structural patterns of errors. For example, in “To-infinitive for modifier/argument of verb,” there were only 22 hits for the pattern, while the number of erroneous triplets is 120. This implies five triplets per hit. This is because, in a deep parser, a wrong choice between adjunct or complement interpretations of a to-infinitival clause affects the interpretation of implicit arguments in the clause through control. Though expected, such detailed observations show how differences between shallow and deep parsers may affect evaluation methods and the methods of analyzing errors.

5.2 Observation of inter-dependencies

In the inter-dependency experiments we performed, some errors could not be forcibly corrected by our method. This was because the parser



Sentence:
The asbestos fiber , crocidolite , is unusually resilient once it enters the lungs , with even brief exposures to it causing symptoms that show up decades later , researchers said .
(a)
(b)
(c)
(d)
Inter-dependent error group (a):
Correct answer: … fiber , : app_2args crocidolite …
ARG1
ARG2
Parser output: … fiber , : coord_2args crocidolite …
Inter-dependent error group (b):
… is usually resilient … the lungs , with …
ARG1
Inter-dependent error group (c):
Correct answer:
… symptoms that show : verb_1arg up decades later …
Parser output:
… symptoms that show : verb_2args up decades later …
Inter-dependent error group (d):
… it causing symptoms that show up decades later …

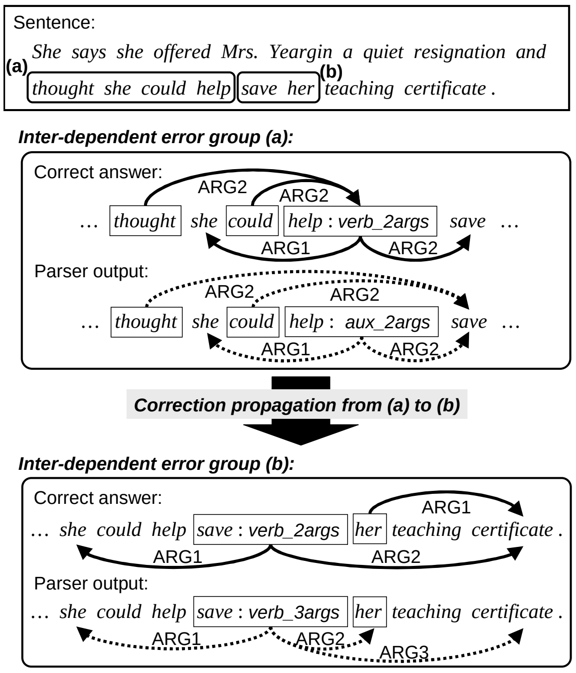
Figure 12: Obtained inter-dependent error groups

we use prunes less probable parse substructures during parsing. In some cases, even if we gave a large positive value to a triplet which should be included in the final parse, parsing paths which can contain the triplet were pruned before. In this research, we ignored such errors as "*uncorrectable*" ones, and focused on the remaining "*correctable*" errors.

Table 4 shows a summary of the analysis. As the previous experiment, Enju produced 4,709 errors for Section 0, of which 3,085 were correctable. By applying the method illustrated in Section 4.2, we obtained 1,978 inter-dependent error groups and 501 correction propagation relationships among the groups.

Figure 11 shows the frequency of the size of inter-dependent error groups. About half of the groups contain only single errors which could have only one-way correction propagations with other errors or were completely independent of other errors.

Figure 12 shows an example of the extracted inter-dependent error groups. For the sentence shown at the top, Enju gave seven errors. By applying the method in Section 4.2, these errors were grouped into four inter-dependent error groups (a) to (d), and no correction propagations were de-



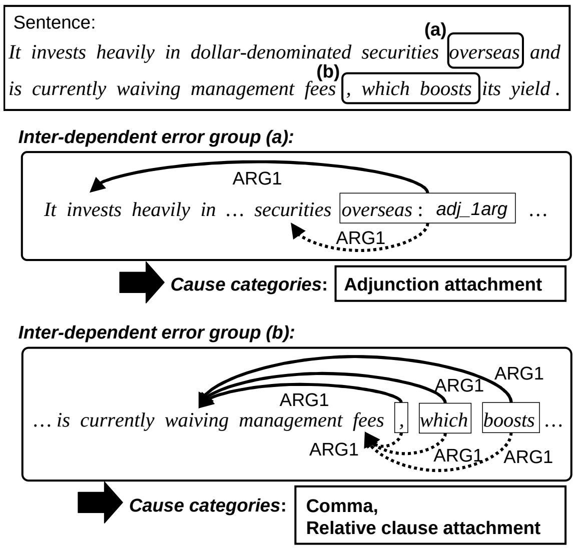
Sentence:
She says she offered Mrs. Yeargin a quiet resignation and thought she could help save her teaching certificate .
(a)
(b)
Inter-dependent error group (a):
Correct answer:
… thought she could help : verb_2args save …
ARG1
ARG2
Parser output:
… thought she could help : aux_2args save …
Correction propagation from (a) to (b)
Inter-dependent error group (b):
Correct answer:
… she could help save : verb_2args her teaching certificate .
Parser output:
… she could help save : verb_3args her teaching certificate .
ARG3

Figure 13: Correction propagation between obtained inter-dependent error groups

tected among them. Group (a) contains two errors on the comma's local behavior as apposition or coordination. Group (b) contains the errors on the words which give almost the same attachment behaviors. Group (c) contains the errors on whether the verb "*show*" took "*decades*" as its object or not. Group (d) contains an error on the attachment of the adverb "*later*". Regardless of the overlap of the regions in the sentence for (c) and (d), our approach successfully grouped the errors into two independent groups. The method shows that the errors in each group are inter-dependent, but errors in one group are independent of those in another. This enables us to concentrate on each of the co-occurring error groups separately, without minding the errors in other groups.

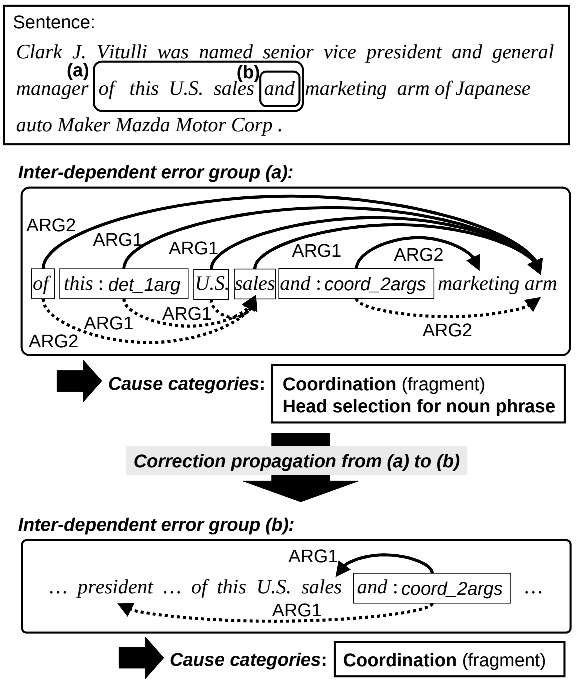
Figure 13 shows another example. In this example, eight errors for a sentence were classified into two inter-dependent error groups (a) and (b). Moreover, it shows that the correction of group (a) results in correction of group (b).

The errors in group (a) were related to the choice as to whether "*help*" had an auxiliary or a pure verbal role. The errors in group (b) were related with the choice as to whether "*save*" took only one object ("*her teaching certificate*") or two objects ("*her*" and "*teaching certificate*"). Between group (a) and (b), no "structural" con-



Sentence:
It invests heavily in dollar-denominated securities overseas and is currently waiving management fees , which boosts its yield .
(a)
(b)
Inter-dependent error group (a):
ARG1
It invests heavily in … securities overseas : adj_1arg …
ARG1
Cause categories: Adjunction attachment
Inter-dependent error group (b):
ARG1
ARG1
ARG1
… is currently waiving management fees , which boosts …
ARG1
ARG1
ARG1
Cause categories: Comma, Relative clause attachment

Figure 14: Combining our two methods (1)



Sentence:
Clark J. Vitulli was named senior vice president and general manager of this U.S. sales and marketing arm of Japanese auto Maker Mazda Motor Corp .
(a)
(b)
Inter-dependent error group (a):
ARG2
ARG1
ARG1
ARG1
ARG2
of this : det_1arg U.S. sales and : coord_2args marketing arm
ARG1
ARG1
ARG2
ARG2
Cause categories: Coordination (fragment) Head selection for noun phrase
Correction propagation from (a) to (b)
Inter-dependent error group (b):
ARG1
… president … of this U.S. sales and : coord_2args …
ARG1
Cause categories: Coordination (fragment)

Figure 15: Combining our two methods (2)

flict could arise when correcting only each of the groups. We could then hypothesize that the correction propagation between the two groups were caused by the disambiguation model.

By dividing the errors into minimal units and clarifying the effects of correcting a target error, we can conclude that the inter-dependent group (a) should be handled first for effective improvement of the parser. In such a way, obtained interdependencies among errors can suggest an effective strategy for parser improvement.

5.3 Combination of the two methods

By combining the two methods described in Section 4.1 and 4.2, we can see how each error cause affects the performance of a parser. The results are summarized in Table 5. The leftmost column in the table shows the numbers of errors in terms of triplets, which are the same as the leftmost column in Table 3.

The "*independence rate*" shows the ratio of erroneous triplets in the category which are not affected by correction of other erroneous triplets. On the other hand, the "*correction effect*" shows how many erroneous triplets would be corrected if one of the erroneous triplets in the category was corrected. These two columns are computed by using the error propagation network constructed in Section 4.2. That is, by using the network we obtain the number of erroneous triplets to be corrected if a given erroneous triplet in the category was corrected, sum up these numbers and then calculate the average number of expected side-effect correction per erroneous triplet in the category.

Figure 14 shows an example of *independent* errors. For the sentence at the top, the parser produced four errors. The method in Section 4.2 successfully discovered two inter-dependent error groups (a) and (b). There was no error propagation relation between the two groups. On the other hand, the method in Section 4.1 associated all of these four errors with the categories of "Adjunction attachment," "Comma" and "Relative clause attachment," and the error for the "Adjunction attachment" corresponds to the inter-dependent error group (a). Because this group is not a receiving node of any propagation in the network, the error is regarded as an "*independent*" one.

Independent errors mean that, if a new parsing model could correct them, the correction would not be destroyed by other errors which remain in the new parsing model.

The *correction effect* shows the opposite and desirable effect of the nature of the dependency among errors which the propagation network represents. This means that, if one of erroneous triplets in the category was corrected, the correction would be amplified through propagation, and as a result other errors would also be corrected.

We show an example of the correction effect in Figure 15. In the figure, the parser had six errors; three false outputs for ARG1 of "*and*," "*this*" and

Cause categories of errors	# of errors	Independence rate (%)	Correction effect (%)	Expected range of error correction
[Argument selection]				
Prepositional attachment	579	74.8	144.3	625.0 - 835.5
Adjunction attachment	261	56.6	179.6	265.3 - 468.8
Conjunction attachment	43	36.4	239.4	37.5 - 102.9
Head selection for noun phrase	30	0.0	381.8	0.0 - 114.5
Coordination	202	42.5	221.2	189.9 - 446.8
[Predicate type selection]				
Preposition/Adjunction	108	41.7	158.3	71.3 - 171.0
Gerund acts as modifier/not	84	46.2	159.0	61.7 - 133.0
Coordination/conjunction	54	44.4	169.4	40.6 - 91.5
# of arguments for preposition	51	95.8	108.3	52.9 - 55.2
Adjunction/adjunctive noun	13	75.0	125.0	12.2 - 16.3
[More structural errors]				
To-infinitive for modifier/argument of verb	120	36.0	116.0	50.1 - 139.2
Subject for passive sentence/not	8	37.5	112.5	3.4 - 9.0
[Others]				
Comma	444	39.5	194.4	341.0 - 863.1
Relative clause attachment	102	32.1	141.7	46.4 - 144.5

Table 5: Correction propagations between errors for each cause category and the other errors

"*U.S.*," two false outputs for ARG2 of "*of*" and "*and*," and a missing output for ARG1 of "*sales*." Our method for inter-dependencies classified these errors into two inter-dependent error groups (a) and (b), and extracted an correction propagation from (a) to (b). Our method for cause categories, on the other hand, associated two errors of "*and*" with the category "Coordination" and one error of "*this*" with the category "Head selection for noun phrase." When we correct an error in the interdependent error group (a), the correction leads to not only correction of the other errors in (a) but also correction of the error in (b) via correction propagation from (a) to (b). Therefore, a correction effect of an error in group (a) results in 6.0.

On the basis of the above considerations, we estimated the range of the effect which an error correction in each category has. The minimum of expected correction range in Table 5 is given by the product of the number of erroneous triplets in the category, the independence rate and the correction effect. On the other hand, the maximum is given by the product of the number of erroneous triplets in the category and the correction effect. This assumes that all corrections made in the category are not cancelled by other errors, while the figure in the minimum are based on the assumption that all corrections made in the category, except for the independent ones, are cancelled by other errors.

Table 5 would thus suggest which categories should be resolved with high priority, from three points of view: the number of errors in the category, the number of independent errors, and the correction effect.

6 Further applications of our methods

In this section, as an example of the further application of our methods, we attempt to analyze parsing behaviors in domain adaptation from the viewpoints of error cause categories.

In Hara et al. (2007), we proposed a method for adapting Enju to a target domain, and then succeeded in improving the parser performance for the GENIA corpus (Kim et al., 2003), a biomedical domain. Table 6 summarizes the parsing results for three types of settings respectively: parsing PTB with Enju ("Enju for PTB"), parsing GENIA with Enju ("Enju for GENIA"), and parsing GENIA with the adapted model ("Adapted for GENIA"). We then analyzed the performance transition among these settings from the viewpoint of the cause categories given in Section 4.1 (Table 7). In order to compare the error frequencies among different settings, we took the percentage of target errors in all of the evaluated triplets. The signed values between the two settings show how much the errors increased when moving from the left settings to the right ones.

When we focus on the transition from "Enju for PTB" to "Enju for GENIA," we can observe that the change in the domain resulted in a different distribution of error causes. The errors for most categories increased, and in particular, the errors for "Prepositional attachment" and "Coordi-

	Enju for PTB	**Enju for GENIA**	**Adapted for GENIA**
Evaluated sentences	1,811	842	842
Evaluated triplets	44,934	22,230	22,230
Errors	4,709	3,120	2,229
F-score (LP/LR)	90.69 (90.78/90.59)	87.41 (87.60/87.22)	90.93 (91.10/90.76)

Table 6: Summary of parsing performances for domain and model variations

Cause categories of errors	**Rate of errors against total examined relations in test set (%)**				
	Enju for PTB	⟶	**Enju for GENIA**	⟶	**Adapted for GENIA**
Classified	**4.62**	+2.60↗	**7.22**	−1.80↘	**5.42**
[Argument selection]					
Prepositional attachment	1.29	+**0.93**↗	2.22	−**0.64**↘	1.58
Adjunction attachment	0.58	+**0.38**↗	0.96	−**0.20**↘	0.76
Conjunction attachment	0.10	−**0.04**↘	0.06	−**0.04**↘	0.02
Head selection for noun phrase	0.07	+**0.17**↗	0.24	−**0.06**↘	0.18
Coordination	0.45	+**0.59**↗	1.04	−**0.25**↘	0.79
[Predicate type selection]					
Preposition/Adjunction	0.24	+**0.08**↗	0.32	−**0.06**↘	0.26
Gerund acts as modifier/not	0.19	−**0.07**↘	0.12	+**0.01**↗	0.13
Coordination/conjunction	0.12	±**0.00**→	0.12	−**0.07**↘	0.05
# of arguments for preposition	0.11	−**0.02**↘	0.09	±**0.00**↘	0.09
Adjunction/adjunctive noun	0.03	+**0.19**↗	0.22	−**0.08**↘	0.14
[More structural errors]					
To-infinitive for modifier/argument of verb	0.27	+**0.02**↗	0.29	−**0.09**↘	0.20
Subject for passive sentence/not	0.02	+**0.34**↗	0.36	+**0.01**↗	0.37
[Others]					
Comma	0.99	−**0.03**↘	0.96	−**0.31**↘	0.65
Relative clause attachment	0.23	+**0.05**↗	0.28	−**0.03**↘	0.25
Unclassified	**5.86**	+**0.96**↗	**6.82**	−**2.22**↘	**4.60**
Total (Classified + Unclassified)	**10.48**	+**3.56**↗	**14.04**	−**4.01**↘	**10.03**

Table 7: Error distributions for domain and model variations

nation" increased remarkably. On the other hand, the transition from "Enju for GENIA" to "Adapted for GENIA" shows that their adaptation method succeeded in reducing the errors for most categories to some extent. However, for "Prepositional attachment," "Coordination," and "Subject for passive sentence or not," there were still noticeable gaps in error distribution between "Enju for PTB" and "Adapted for GENIA." This would mean that the adapted model requires further performance improvement if we expect the same level of performances for those categories as the parser originally obtained in PTB.

We could thus capture some biases of cause categories which occur in domain transition or in domain adaptation, which would not be clarified by F-score evaluation methods. With inter-dependencies given by the method described in Section 4.2, the above analysis would be useful for effectively exploring further adaptation.

7 Related works

Although there have been many researchers who analyzed errors in their own systems in the experiments, there has been little research which focused on error analysis itself.

In the field of parsing, McDonald and Nivre (2007) compared parsing errors between graph-based and transition-based parsers. They considered accuracy transitions from various points of view, and the obtained statistical data suggested that error propagation seemed to occur in the graph structures of parsing outputs. Our research proceeded one step further and attempted to reveal the nature of the propagations. In examining the combination of the two types of parsing, they utilized approaches similar to our method for capturing inter-dependencies of errors. They allowed a parser to give only structures produced by the parsers and utilized the ideas for evaluating the parser's potentials, whereas we utilized it for observing error propagations.

Dredze et al. (2007) showed that many of the parsing errors in domain adaptation tasks may come from inconsistencies between the annotations of training resources. This would suggest that just error comparisons without considering the inconsistencies could lead to a misunderstanding of what happens in domain transitions. The summarized error cause categories and inter-dependencies given by our methods would be useful clues for extracting such domain-dependent error phenomena.

When we look into other research areas in natural language processing, Giménez and Màrquez (2008) proposed an automatic error analysis approach in machine translation (MT) technologies. They developed a metric set which could capture features in MT outputs at different linguistic levels with different levels of granularity. Like we considered parsing systems, they explored ways to resolve costly and rewardless error analysis in the MT field. One of their objectives was to enable researchers to easily obtain detailed linguistic reports on the behavior of their systems, and to concentrate on analyses for the system improvements.

8 Conclusion

We proposed two methods for analyzing parsing errors. One is to assign errors to cause categories, and the other is to capture inter-dependencies among errors. The first method defines error patterns to identify cause categories and then associates errors involved in the patterns with the corresponding categories. The second method re-parses a sentence with a target error corrected, and regards errors corrected together as dependent on the target.

In our experiments with an HPSG parser, we successfully associated more than 40% of the errors with 14 cause categories, and captured 1,978 inter-dependent error groups. Moreover, the combination of our methods gave a more detailed error analysis for effective improvement of the parser.

In our future work, we would give more pattern rules for classifying a large percentage of errors into cause categories, and incorporate *uncorrectable* errors into inter-dependency analysis. After improving the analytical facilities of our individual methods, we would explore the possibility of combining the methods for obtaining more powerful and detailed clues on how to improve parsing performance.

Acknowledgments

This work was partially supported by Grant-in-Aid for Specially Promoted Research (MEXT, Japan).

References

Mark Dredze, John Blitzer, Partha Pratim Talukdar, Kuzman Ganchev, João V. Graça, and Fernando Pereira. 2007. Frustratingly hard domain adaptation for dependency parsing. In *Proceedings of the CoNLL Shared Task Session of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 1051–1055.

Jesús Giménez and Lluís Màrquez. 2008. Towards heterogeneous automatic mt error analysis. In *Proceedings of the Sixth International Language Resources and Evaluation (LREC'08)*, pages 1894–1901.

Tadayoshi Hara, Yusuke Miyao, and Jun'ichi Tsujii. 2007. Evaluating impact of re-training a lexical disambiguation model on domain adaptation of an hpsg parser. In *Proceedings of 10th International Conference on Parsing Technologies (IWPT 2007)*, pages 11–22.

Ronald M. Kaplan and Joan Bresnan. 1995. Lexical-functional grammar: A formal system for grammatical representation. *Formal Issues in Lexical-Functional Grammar*, pages 29–130.

Jin-Dong Kim, Tomoko Ohta, Yuka Teteisi, and Jun'ichi Tsujii. 2003. GENIA corpus - a semantically annotated corpus for bio-textmining. *Bioinformatics*, 19(suppl. 1):i180–i182.

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 *Proceedings of ARPA Human Language Technology Workshop*.

Ryan McDonald and Joakim Nivre. 2007. Characterizing the errors of data-driven dependency parsing models. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 122–131.

Yusuke Miyao and Jun'ichi Tsujii. 2005. Probabilistic disambiguation models for wide-coverage HPSG parsing. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics (ACL)*, pages 83–90.

Takashi Ninomiya, Takuya Matsuzaki, Yoshimasa Tsuruoka, Yusuke Miyao, and Jun'ichi Tsujii. 2006. Extremely lexicalized models for accurate and fast HPSG parsing. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 155–163.

Carl J. Pollard and Ivan A. Sag. 1994. *Head-Driven Phrase Structure Grammar*. University of Chicago Press.

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