ANALYZING COORDINATE STRUCTURES INCLUDING PUNCTUATION IN ENGLISH

Sadao Kurohashi*
Section of Electronics and Communication, Kyoto University
Yoshida-honmachi, Sakyo, Kyoto, 606-01 Japan
kuro@kuee.kyoto-u.ac.jp

Abstruct

We present a method of identifying coordinate structure scopes and determining usages of commas in sentences at the same time. All possible interpretations concerning comma usages and coordinate structure scopes are ranked by taking advantage of parallelism between conjoined phrases/clauses/sentences and calculating their similarity scores. We evaluated this method through experiments on held-out test sentences and obtained promising results: both the success ratio of interpreting commas and that of detecting CS scopes were about 80%.

1 Introduction

It is commonly recognized that coordinate structures (CSs) present both linguistic and practical difficulties [Steedman. 1990, Agarwal and Boggess, 1992, Okumura and Muraki, 1994]. The practical difficulties are especially serious because not only do CSs themselves have complex scope ambiguity, but also sentences with CSs tend to be long and the combination of scope ambiguity and the intrinsic syntactic ambiguity in long sentences can easily cause combinatorial explosion.

We first noticed the importance of solving this problem in the analysis of Japanese sentences, and developed a unique, efficient method of resolving CS scope ambiguities [Kurohashi and Nagao, 1992]. The underlying assumption of this method is that conjoined phrases/clauses/sentences exhibit parallelism, that is, they have a certain similarity in the sequence of words and their grammatical structures as a whole. Based on this assumption, we devised an algorithm which determines conjuncts by finding the two most similar word-sequences on the left and the right of a conjunction (this method is called *similarity-based CS analysis*). By incorporating this method into a conventional syntactic parser, we developed a reasonably good parsing system which can work on real world texts [Kurohashi and Nagao, 1994].

Since we expected that the analysis based on this assumption would also work well for English CSs, we slightly modified our system to handle English sentences and tested it against a certain amount of text. We found that it works well on simple cases, like sentences containing only one conjunction, but it does not always work well on real sentences. In Japanese, it is morphologically clear what kind of expressions indicate the existence of CSs and whether a CS consists of noun phrases or verb phrases. Therefore, even if two or more CSs exist in a Japanese sentence, it is not so hard to order their scopes and to get a totally consistent structure. In

*This work was done at Institute for Research in Cognitive Science, University of Pennsylvania. We would like to thank Prof. Aravind K. Joshi, Prof. Mitchell P. Marcus, B Srinivas, and Dania Egedi for helpful comments and advice.

English, however, it is not morphologically clear and so the analysis of many CSs in an English sentence is much harder. In particular, commas cause serious problems since they have a variety of usages: some of them indicate CSs, some surround parenthetical expressions, and others just indicate constituent boundaries.

To solve this problem, we have developed the following method:

1. enumerate all possible interpretations concerning usages of commas and hierarchical structures between CSs (this interpretation is called *general structure* (GS)),
2. for each GS, detect the most plausible scopes of the CSs using the similarity-based CS analysis (here, each CS is given a score expressing the similarity between its conjuncts),
3. rank the GSs roughly according to the sum of the similarity scores for the CSs in each GS. This results in the final analysis.

Another possible solution is to parse a sentence and then check parallelism of parsed possible CSs. This method has advantages in that the structural information in the CSs is available and checking syntactically impossible scopes can be avoided. In practice, however, it is very difficult to employ this method because sentences containing CSs tend to be long and an unacceptably large number of parses are often produced for such long sentences.[1] Our method presented here also produces an unacceptably large number of GSs for some sentences, however, the combinatorial explosion of GSs is much slower than that of straight parsing, so most sentences containing CSs can be handled.

Although some of the recent work on English CSs has taken their parallelism into account [Okumura and Muraki, 1994], there is no work that we are aware of which resolves the inherent problems of handling CSs — the dual problems of handling many CSs in a sentence and of interpreting commas.

We start by introducing the similarity-based CS analysis in Section 2. Then we clarify the necessity of enumerating possible GSs, and present our algorithm in Section 3. Finally we discuss the experimental evaluation in Section 4.

2 Similarity-Based CS Analysis

First of all, we describe the core engine of our method, similarity-based CS analysis, which takes advantage of parallelism between conjoined phrases/clauses/sentences. This method tries to find the two most similar word-sequences on the left and the right of a conjunction, and then considers them conjuncts.

The following explains the definition of a similarity measure between two word sequences and how to calculate it (Figure 3 to 5 are concrete examples, and the adjusted parameters used in our experiment are shown in the Appendix).

- An input sentence should be in the following form:

  $$w_0/POS_0 \;\; k_1 \;\; w_1/POS_1 \;\; k_2 \;\; w_2/POS_2 \;\; \ldots \;\; k_l \;\; w_l/POS_l$$

  where w_i is a word other than *and*, *or*, *but*, comma(,), POS_i is one or more possible parts-of-speech (POS) of w_i, k_i is a sequence of zero or more *and*, *or*, *but*, comma(,). Hereafter, the possible values of k_i, such as ",", "*and*", ", *and*", are called *key expressions*.

[1]Furthermore, part-of-speech tagging near conjunctions is not very reliable and CSs sometimes contain gaps, so it is intrinsically difficult to parse sentences containing CSs.

- A triangular matrix, $A = (a(i,j))$, is composed for an input sentence, where the i-th diagonal element is $k_i w_i$, and another element $a(i,j)$ $(i < j)$ is the similarity value between words w_i and w_j. A similarity value between two words is calculated according to POS matching, exact word matching, and their closeness in a thesaurus tree (WordNet is currently being used [Beckwith *et al.*, 1991]).
- In detecting a CS scope concerning k_n, a *path* is defined as follows, which basically shows correspondences between words in candidate pre- and post-conjuncts:

  path ::= $(a(p_1, q_1), a(p_2, q_2), \ldots, a(p_r, q_r))$

  where $q_1 = n,\ p_i < p_{i+1},\ q_i < q_{i+1},\ p_r = n - 1$.
- A *path score* is defined by the following five criteria which indicate the similarity between two word sequences on the left side of the path and under the path.
  - sum of each element's points on the path,
  - penalty points when the path extends non-diagonally (which causes unbalanced lengths of conjuncts),
  - penalty points if the path contains symbols indicating a constituent boundary, such as a comma,
  - bonus points on expressions signaling the beginning or ending of a CS, such as *both*,
  - normalizing the total score of the above four criteria by the n-th power $(0 \leq n \leq 1)$ of the number of words covered by the path $(w_{p_1} \ldots w_{q_r})$.
- When the path with the maximum path score concerning the key expression, k_n, is detected, two word sequences on the left side of the path and under the path are considered to be the two most similar word sequences, that is, to be the conjuncts concerning k_n. Calculation of path scores and selection of the path with the maximum path score are done using a dynamic programming method (see [Kurohashi and Nagao, 1992] for the details).

3 Enhanced Similarity-Based CS Analysis

3.1 Necessity of Enumerating Possible General Structures

The similarity-based CS analysis just described can identify a CS scope for two conjuncts on the left and the right of a key expression. However, in real sentences there often exist CSs consisting of three or more conjuncts, such as "*A, B, and C*", and some sentences may contain two or more CSs (which can be hierarchical). In order to handle these cases, the original method for Japanese sentences consisted of the following steps:

1. for all key expressions in a sentence, detect their CS scopes,
2. by checking overlapping relations in all pairs of detected CS scopes,
   - combine CSs into one CS whenever they consecutively overlap each other (i.e., "*A, B, and C*" is first detected as "*A, B*" and "*B and C*"),
   - recognize a hierarchical relation between two CSs whenever one of them includes the other entirely.

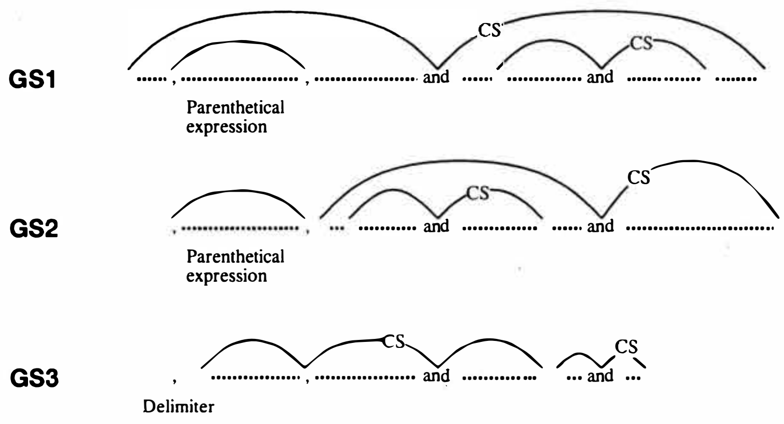

Figure 1: Examples of general structures.

In English, however, this method of processing complex CSs does not always work well. For example, a sentence like " ... , ... , ... *and* ... *and* ..." has a number of possible general structures (GSs), that is, a number of possible interpretations concerning usages of commas and hierarchical structures between the CSs. Some of these are shown in Figure 1. Suppose the first GS in Figure 1 is correct. When the similarity scores between two word sequences around the first *and* are calculated on the original sentence, it is possible that correct conjuncts do not have the maximum similarity score. Instead, an incorrect scope, like the second GS in Figure 1, may have a larger similarity score and thus be selected.

In order to give a reasonably large similarity score to the correct conjuncts, the following steps are necessary: 1) postulate that the first GS is correct, that is, the two commas surround a parenthetical expression and the CS concerning the first *and* contains the CS concerning the second *and*, 2) assume a reduced sentence in which the parenthetical expression and either conjunct of the latter CS are removed, and 3) perform the similarity-based CS analysis on the reduced sentence. Of course, since we do not know which GS is correct, this test must be performed on all possible GSs.

3.2 Enhanced Similarity-Based CS Analysis

Based on the considerations detailed in the previous section, we have developed an enhanced CS analysis method to process English sentences with complex CSs. The steps of this enhanced method are presented in detail below.

Step 1. Enumerating Possible GSs

The main problem in CS analysis is the ambiguity of commas. A comma may be used in any of the following ways.

Strong connector — followed by *and* or *or*, resulting in a CS with three or more conjuncts,

Weak connector — not followed by *and* or *or* but resulting a CS,

Parenthesizor — surrounding a parenthetical expression with another comma,

Delimiter — indicating a certain constituent boundary.

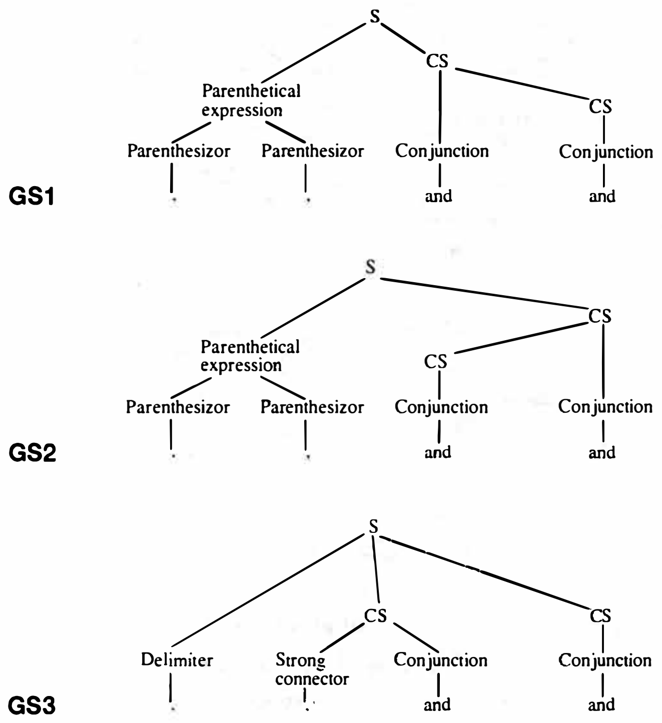

Figure 2: Examples of parse trees of key expressions.

Possible GSs are created by combining these interpretations of commas with patterns of hierarchical structures among CSs and parenthetical expressions. We constructed a set of phrase structure rules which regards key expressions[2] as terminal symbols and translates a sequence of key expressions to possible GSs. Some parse trees resulting from the sequence of key expressions, “ ... , ... , ... *and* ... *and* ...”, and corresponding to the GSs in Figure 1 are shown in Figure 2. This sequence of key expressions has 135 possible parses (GSs) in total.

Step 2. Detecting Plausible CS Scopes for each GS

Then, for each GS, CSs are analyzed, starting with the innermost CS in the GS and working outward. For each CS, its possible scope is limited to retain the GS, and the scope with the maximum similarity score within the limits is detected by the similarity-based CS analysis. For example, in the case of the first GS in Figure 1, a CS scope for the right *and* is first detected on the condition that its starting point is restricted not to go over the left *and*. Then a CS scope for the left *and* is detected within the range of its post-conjunct including the left entire CS.

Once the scope of a CS is detected, its conjuncts other than the first one are removed. In the case of a parenthetical expression, its content is removed after analyzing all CSs within it. By this process of reducing a sentence, the similarity-based CS analysis for an upper level CS in a GS which contains CSs and/or parenthetical expressions can be done more precisely and its proper scope is likely to be detected.

[2]While handling only commas as a punctuation symbol now, we intend to extend our system to handle other symbols, such as hyphen(-), colon(:), semicolon(;).

Step 3. Ranking Possible GSs

Scores are given to key expressions according to their usage. For conjunctions and connectors (commas) of CSs, the scores are based on the similarity score between their conjuncts, while for delimiters and parenthesizors, they are based on the expressions around them. As a final result, we rank all the possible GSs according to the sum of scores for all key expressions in each GS.

To give decreasing priority to the evaluation of a CS indicated by a conjunction, a strong connector, and a weak connector, the similarity scores of their CSs are weighted by $w_{conjunction}$, $w_{s\_connector}$, $w_{w\_connector}$ respectively, where $w_{conjunction} > w_{s\_connector} > w_{w\_connector}$.

When a key expression is a comma interpreted as a delimiter or a parenthesizor, a larger score is given if it appears with an expression being considered to be parenthetical or calling for a delimiter. If not, a smaller score is given. For example, when the commas in a sentence, "... , followed by ... , ...", are interpreted as a parenthesizor, they are given larger scores.

4 Experiments and Discussion

We report the experimental results to illustrate the effectiveness of our present method. From the beginning of our work on English CSs, we have used an article from Brown Corpus to test the original method, to develop the enhanced similarity-based CS analysis method, and to adjust the various parameters in the method (the adjusted parameters are shown in the Appendix).[3] Then we randomly chose six more articles from Brown Corpus, and analyzed them using the method with the adjusted parameters.

4.1 Experimental Evaluation

Table 1 shows the statistics of sentences with key expressions and the average number of their possible GSs. Out of 709 sentences of the six test articles, 424 sentences contain commas and/or conjunctions and need this kind of analysis. Among them, 14 sentences which contained six or more key expressions and would produce an unacceptably large number of GSs (underlined in Table 1) were excluded from the experimentation.[4]

An input to our method is a POS tagged sentence. The following four types of test sets were prepared:

Type 1 : manually tagged (revised) sentences found in Penn Treebank [Marcus *et al.*, 1993],

Type 2 : one-best tagged sentences by Brill's tagger [Brill, 1994],

Type 3 : n-best tagged sentences by Brill's tagger,

Type 4 : combination of the one-best and n-best tagged sentences (n-best tags were given only to words adjoining key expressions).

[3]The parameters used in this paper were adjusted manually through experimentation. As for automatic parameter tuning, we are currently working on a project, getting promising results. This issue will be published in the near future.

[4]In sentences containing six or more key expressions, some of the key expressions often compose very simple CSs, like "... houses, hotels, clubs, ships, and theaters ..." or "... red, green, or blue ... ". By constructing heuristic rules for such simple CSs and applying them to input sentences first, the number of possible GSs would be reduced so that our method can handle sentences with many key expressions. Implementation of such mechanism will be a target of our future work.

Table 1: Statistics of key expressions and GSs.

# of *and*, *or*, *but*	# of commas							
	0	1	2	3	4	5	6	7
0	285 (–)	59 (2)	29 (8)	6 (44)	3 (284)	2 (2004)	1 (?)	0 (–)
1	137 (1)	64 (5)	25 (29)	5 (182)	1 (1529)	2 (?)	0 (–)	1 (?)
2	32 (3)	18 (18)	9 (119)	3 (932)	3 (?)	0 (–)	0 (–)	0 (–)
3	5 (12)	6 (82)	4 (574)	1 (?)	1 (?)	1 (?)	0 (–)	1 (?)
4	0 (–)	2 (416)	1 (?)	0 (–)	0 (–)	0 (–)	0 (–)	0 (–)
5	0 (–)	0 (–)	2 (?)	0 (–)	0 (–)	0 (–)	0 (–)	0 (–)

a (b)

a : # of sentences occurring in the test articles,
b : average # of GSs.

Table 2: Analysis results of interpreting commas and detecting CS scopes.

	Comma interpretation	CS scope detection
Type 1 (with Penn Treebank tags)	83.8%	81.0%
Type 2 (with one-best tags)	82.6%	79.0%
Type 3 (with n-best tags)	72.9%	67.7%
Type 4 (with combined tags)	83.2%	77.6%

Table 3: Analysis results of interpreting commas.

Correct interpretation	Detected interpretation		
	Strong connectors	Weak connectors	Parenthesizors or delimiters
Strong connectors	35	11	8
Weak connectors	5	28	6
Parenthesizors or delimiters	6	23	229

The total success ratio was 83.2% ((35+28+229)/(35+11+8+5+28+6+6+23+229)).

Table 4: Analysis results of detecting CS scopes.

Types of key expressions	*,*	*; ;*	*, , ,*	*and* *or* *but*	*, and* *, or*	*, , and* *, , or*	*, , , and* *, , , or*	Total
# of correct analysis	17	2	2	286	26	3	1	337 (77.6%)
# of occurrence	22	4	3	356	40	8	1	434

We analyzed these test sets by our method and checked the results as follows:

Interpretation of commas

We checked whether commas were interpreted correctly or not (the left-hand side of Table 2).

Detection of CS scopes

We also checked whether the detection of a CS scope was correct or not by automatically comparing the analysis results with the parsed sentences in the Penn Treebank[5] (the right-hand side of Table 2). Since the aim of our method is to get rough information on CS scopes (mentioned in the next section in detail), we regarded a detected scope as correct when the detected CS scope contains at least the head words of all conjuncts and, if any, at most one extra constituent of determiner or preposition.

The success ratios both for the comma interpretation and for the CS scope detection were around 80%, for the test sets of type 1, 2 and 4. We concluded that they were fairly good, comparing the results of previous work [Agarwal and Boggess, 1992, Okumura and Muraki, 1994] and considering that our method handles the comma interpretation as well as the CS scope detection. Table 3 and 4 show the details of our evaluation for the combined-tag test set (type 4).

4.2 Discussion

Goal of Our Method

Here, we would like to make the aim of our method clear. Our goal is to get a general information on sentences containing commas and/or CSs, including the interpretation of comma usages and the scopes of CSs. Although the CS scopes obtained by our method are only rough approximations, to get such rough approximations is the first and most critical task. Later stages can be used to determine their exact scopes.

For example, suppose there is an input sentence whose POS sequence is "N_1 N_2 V_1 N_3 N_4 *and* N_5 V_2 N_6 N_7". The aim of our method is to determine whether the *and* conjoins two sentences or two noun phrases. If our method estimates that the two sentences are conjoined, the detected scope might not contain border nouns, such as N_1 and N_7. But they would be easily included in the CS scope at a subsequent syntactic analysis stage. On the other hand, when our method estimates the noun phrase coordination, it might not return the precise scope, either. The two possibilities, "(N_3 (N_4 and N_5))" or "((N_3 N_4) and N_5)", would need to be distinguished at a semantic analysis stage.

POS Tagging and CS Analysis

Since POS tagging near conjunctions is not very reliable, n-best tagged sentences, rather than one-best tagged sentences, are considered to be a better input to a CS analysis system, simulating its real application. The current n-best tagger, however, assigns too many tags to the words. As a result, it causes a troublesome side-effect, giving inappropriate similarity scores to many pairs of words, and the success ratio of CS scope detection for n-best tagged sentences becomes very low as shown in Table 2.

[5]Since CSs by weak connectors were not explicitly indicated in the Penn Treebank, they were checked manually.

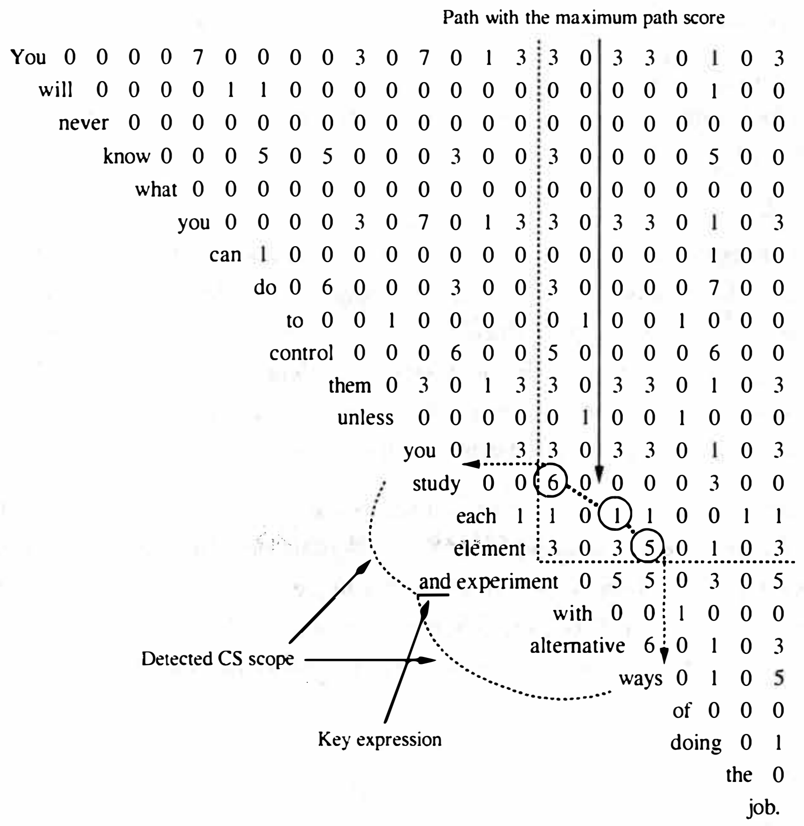
Path with the maximum path score
You 0 0 0 0 7 0 0 0 0 3 0 7 0 1 3 3 0 3 3 0 1 0 3
will 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0
never 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
know 0 0 0 5 0 5 0 0 0 3 0 0 3 0 0 0 0 5 0 0
what 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
you 0 0 0 0 3 0 7 0 1 3 3 0 3 3 0 1 0 3
can 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0
do 0 6 0 0 0 3 0 0 3 0 0 0 0 7 0 0
to 0 0 1 0 0 0 0 0 1 0 0 1 0 0 0
control 0 0 0 6 0 0 5 0 0 0 0 6 0 0
them 0 3 0 1 3 3 0 3 3 0 1 0 3
unless 0 0 0 0 0 1 0 0 1 0 0 0
you 0 1 3 3 0 3 3 0 1 0 3
study 0 0 6 0 0 0 0 3 0 0
each 1 1 0 1 1 0 0 1 1
element 3 0 3 5 0 1 0 3
and experiment 0 5 5 0 3 0 5
with 0 0 1 0 0 0
alternative 6 0 1 0 3
ways 0 1 0 5
of 0 0 0
doing 0 1
the 0
job.
Detected CS scope
Key expression

Figure 3: An example of detecting CS scopes and interpreting comma usages (1).

One compromise we have done is that we combined the results by the one-best tagger and that by the n-best tagger, as in our test set, type 4. For this test set, the success ratio was as good as the one-best tagged set, and some CSs which could not be detected correctly because of the errors of the one-best tagger, were detected correctly by using the combined tags (Figure 3 shows an example of such CSs).

This result is very promising since it shows that our method can handle two very difficult problems: POS ambiguity and CS scope ambiguity, simultaneously in a sense. (Note that getting the correct CS scope directly leads to the resolution of POS ambiguity around the key expression.)

Examples of Correct Analysis

In the case of the example in Figure 3, the verb phrase coordination was analyzed correctly. The detected scope of the CS was regarded as correct since the two head verbs are included, although the ending words are not. In this example, when the one-best tagger was used, the incorrect tag, **NN** (Noun), was given to the head verb of the pre-conjunct ("experiment") and so the CS could not be analyzed correctly. On the other hand, the use of the n-best tags assignment to the word, **NN** (Noun), **VB** (Verb, base form), and **VBP** (Verb, non-3rd person, singular, present) led to the correct analysis, as shown in Figure 3.

In the case of the example sentence in Figure 4, the two commas were correctly interpreted as surrounding a parenthetical expression, and the verb phrase coordination was analyzed correctly.

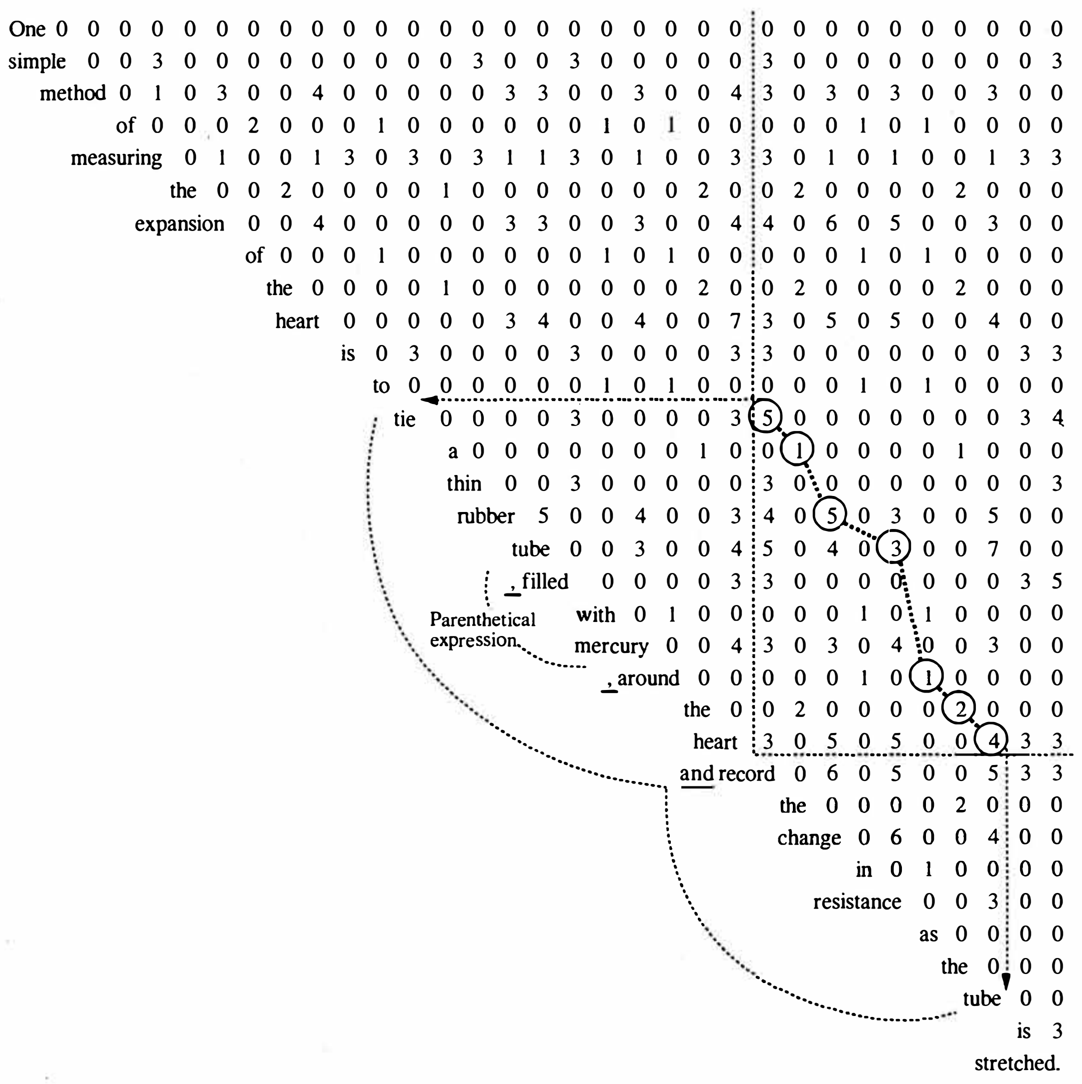

Figure 4: An example of detecting CS scopes and interpreting comma usages (2).

In the example in Figure 5, the sentential coordination containing a gap and indicated by a weak connector was analyzed correctly. The former comma is correctly interpreted as a delimiter; the latter as a weak connector.

Types of Incorrect Analysis

Finally, we can classify the types of incorrect analyses produced by our method.

- Some types of CSs are extremely hard to handle. The following CS from a test sentence consists of a prepositional phrase and wh-phrase, and it cannot be detected on a parallel basis (angle brackets are inserted to indicate the correct CS scope):

  ...but anyone ⟨ with a wide acquaintance in both groups ⟩ and ⟨ who has sat through the many round tables ⟩ will recognize ...

  Such expressions are, however, very rare in actual texts.

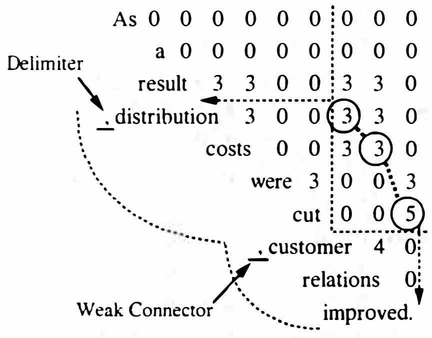

Figure 5: An example of detecting CS scopes and interpreting comma usages (3).

- Some idiomatic expressions were not analyzed correctly, for example, "*may and more likely may not*", "*... or nothing at all*". There seems to be no solution but to prepare a set of specific rules for such expressions. Since parsed corpora are now widely available, we believe that it will not be difficult to collect such expressions.
- As for the interpretation of commas, it was difficult to discriminate weak connectors from others. The comma in the following sentence was incorrectly interpreted as a weak connector (angle brackets indicate the incorrectly detected CS scope):

  Outputs of the two systems are ⟨ measured by a pulse-timing circuit and a resistance bridge ⟩ , ⟨ followed by a simple analogue computer which feeds a multichannel recorder. ⟩

  We need to establish more precise detection methods for weak connectors.

5 Conclusion

We have proposed a method of detecting valuable information from real sentences which contain commas and/or CSs, including the interpretation of comma usages and the scopes of CSs. The core engine of our method is a way of identifying conjuncts by checking their parallelism. Using this engine, we have developed a method of enumerating possible general structures of a sentence and ranking them based on similarity scores for their CSs.

The information obtained by this method can be used to guide a syntactic parser to obtain parses for actual, long sentences. Integrating this method with an existing parsing system will be our next target.

References

[Agarwal and Boggess, 1992] Rajeev Agarwal and Lois Boggess. A simple but useful approach to conjunct identification. In *30th Annual Meeting of the Association for Computational Linguistics*, pages 15–21, 1992.

[Beckwith *et al.*, 1991] Richard Beckwith, Christiane Fellbaum, Derek Gross, and George Miller. WordNet: A lexical database organized on psycholinguistic principles. In Uri Zernik, editors, *Lexical Acquisition: Exploiting On-Line Resources to Build a Lexicon*, pages 211–232, Erlbaum, 1991.

[Brill, 1994] Eric Brill. Some advances in rule-based part of speech tagging. In *Proceedings of the Twelfth National Conference on Artificial Intelligence (AAAI-94)*, Seattle, 1994.

[Kurohashi and Nagao, 1992] Sadao Kurohashi and Makoto Nagao. Dynamic programming method for analyzing conjunctive structures in Japanese. In *Proceedings of 14th COLING*, Vol.1, pages 170–176, Nantes, 1992.

[Kurohashi and Nagao, 1994] Sadao Kurohashi and Makoto Nagao. A syntactic analysis method of long Japanese sentences based on the detection of conjunctive structures. *Computational Linguistics*, 20(4):507–534, 1994.

[Marcus *et al.*, 1993] Mitchell P. Marcus, Beatrice Santorini and Mary Ann Marcinkiewicz. Building a large annotated corpus of English: the Penn Treebank. *Computational Linguistics*, 19(2):313–330, 1993.

[Okumura and Muraki, 1994] Akitoshi Okumura and Kazunori Muraki. Symmetric pattern matching analysis for English coordinate structures. In *Proceedings of the 4th Conference on Applied Natural Language Processing*, pages 41–46, Stuttgart, 1994.

[Steedman, 1990] Mark J. Steedman. Gapping as constituent coordination. *Linguistics and Philosophy*, 13:207–263, 1990.

Appendix : Parameters

Similarity value between two words	
POS matching of content words	3
POS matching of functional words	1
Exact word matching of content words	4
Exact word matching of functional words	1
Semantic similarity by WordNet	(8 − two words' distance in WordNet)/2
Path Score	
penalty on non-diagonal path extension	
causing to skip a content word	−2
causing to skip a functional word	−1
Penalty on containing a comma being a delimiter	−4
Bonus point on CS starting expressions	
between, *both* (← *and*)	3
either, *neither whether*(← *or*)	3
not only (← *but also*)	10
The power of normalization	2/3
GS ranking	
Weights for key expressions of CSs	
$w_{conjunction}$	1.5
$w_{s\_connector}$	1.0
$w_{w\_connector}$	0.5
Score for a parenthesizor	1.8 or 0.6 (depending on neighbor expressions)
Score for a delimiter	1.79 or 0.59 (depending on neighbor expressions)