Correcting Errors in a Treebank Based on Synchronous Tree Substitution Grammar

Yoshihide Kato[1] and **Shigeki Matsubara**[2]
[1]Information Technology Center, Nagoya University
[2]Graduate School of Information Science, Nagoya University
Furo-cho, Chikusa-ku, Nagoya, 464-8601 Japan
yosihide@el.itc.nagoya-u.ac.jp

Abstract

This paper proposes a method of correcting annotation errors in a treebank. By using a synchronous grammar, the method transforms parse trees containing annotation errors into the ones whose errors are corrected. The synchronous grammar is automatically induced from the treebank. We report an experimental result of applying our method to the Penn Treebank. The result demonstrates that our method corrects syntactic annotation errors with high precision.

1 Introduction

Annotated corpora play an important role in the fields such as theoretical linguistic researches or the development of NLP systems. However, they often contain annotation errors which are caused by a manual or semi-manual mark-up process. These errors are problematic for corpus-based researches.

To solve this problem, several error detection and correction methods have been proposed so far (Eskin, 2000; Nakagawa and Matsumoto, 2002; Dickinson and Meurers, 2003a; Dickinson and Meurers, 2003b; Ule and Simov, 2004; Murata et al., 2005; Dickinson and Meurers, 2005; Boyd et al., 2008). These methods detect corpus positions which are marked up incorrectly, and find the correct labels (e.g. pos-tags) for those positions. However, the methods cannot correct errors in structural annotation. This means that they are insufficient to correct annotation errors in a treebank.

This paper proposes a method of correcting errors in structural annotation. Our method is based on a synchronous grammar formalism, called *synchronous tree substitution grammar* (STSG) (Eisner, 2003), which defines a tree-to-tree transformation. By using an STSG, our method transforms parse trees containing errors into the ones whose errors are corrected. The grammar is automatically induced from the treebank. To select STSG rules which are useful for error correction, we define a score function based on the occurrence frequencies of the rules. An experimental result shows that the selected rules archive high precision.

This paper is organized as follows: Section 2 gives an overview of previous work. Section 3 explains our method of correcting errors in a treebank. Section 4 reports an experimental result using the Penn Treebank.

2 Previous Work

This section summarizes previous methods for correcting errors in corpus annotation and discusses their problem.

Some research addresses the detection of errors in pos-annotation (Nakagawa and Matsumoto, 2002; Dickinson and Meurers, 2003a), syntactic annotation (Dickinson and Meurers, 2003b; Ule and Simov, 2004; Dickinson and Meurers, 2005), and dependency annotation (Boyd et al., 2008). These methods only detect corpus positions where errors occur. It is unclear how we can correct the errors.

Several methods can correct annotation errors (Eskin, 2000; Murata et al., 2005). These methods are to correct tag-annotation errors, that is, they simply suggest a candidate tag for each position where an error is detected. The methods cannot correct syntactic annotation errors, because syntactic annotation is structural. There is no approach to correct structural annotation errors.

To clarify the problem, let us consider an example. Figure 1 depicts two parse trees annotated according to the Penn Treebank annotation [1]. The

[1] 0 and ⋆T⋆ are null elements.

Proceedings of the ACL 2010 Conference Short Papers, pages 74–79,
Uppsala, Sweden, 11-16 July 2010.

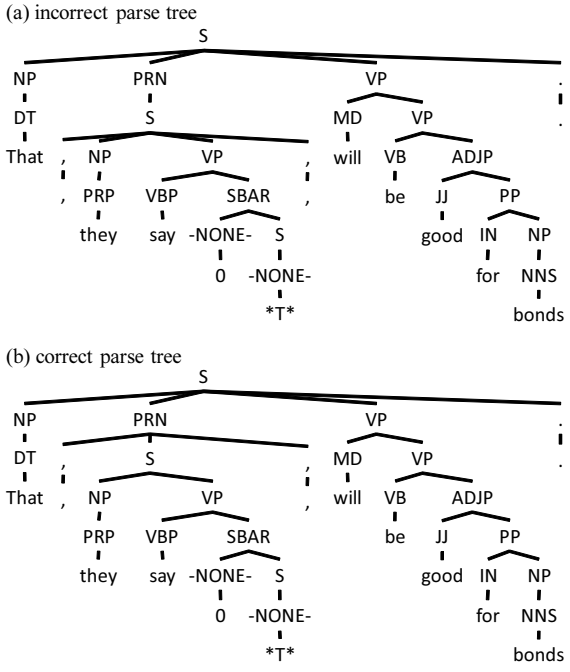

Figure 1: An example of a treebank error

parse tree (a) contains errors and the parse tree (b) is the corrected version. In the parse tree (a), the positions of the two subtrees (`, ,`) are erroneous. To correct the errors, we need to move the subtrees to the positions which are directly dominated by the node `PRN`. This example demonstrates that we need a framework of transforming tree structures to correct structural annotation errors.

3 Correcting Errors by Using Synchronous Grammar

To solve the problem described in Section 2, this section proposes a method of correcting structural annotation errors by using a synchronous tree substitution grammar (STSG) (Eisner, 2003). An STSG defines a tree-to-tree transformation. Our method induces an STSG which transforms parse trees containing errors into the ones whose errors are corrected.

3.1 Synchronous Tree Substitution Grammar

First of all, we describe the STSG formalism. An STSG defines a set of tree pairs. An STSG can be treated as a tree transducer which takes a tree as input and produces a tree as output. Each grammar rule consists of the following elements:

- a pair of trees called *elementary trees*

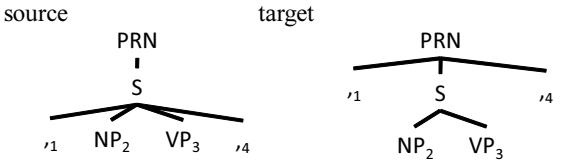

Figure 2: An example of an STSG rule

- a one-to-one alignment between nodes in the elementary trees

For a tree pair $\langle t, t' \rangle$, the tree t and t' are called *source* and *target*, respectively. The nonterminal leaves of elementary trees are called *frontier nodes*. There exists a one-to-one alignment between the frontier nodes in t and t'. The rule means that the structure which matches the source elementary tree is transformed into the structure which is represented by the target elementary tree. Figure 2 shows an example of an STSG rule. The subscripts indicate the alignment. This rule can correct the errors in the parse tree (a) depicted in Figure 1.

An STSG derives tree pairs. Any derivation process starts with the pair of nodes labeled with special symbols called *start symbols*. A derivation proceeds in the following steps:

1. Choose a pair of frontier nodes $\langle \eta, \eta' \rangle$ for which there exists an alignment.
2. Choose a rule $\langle t, t' \rangle$ s.t. $label(\eta) = root(t)$ and $label(\eta') = root(t')$ where $label(\eta)$ is the label of η and $root(t)$ is the root label of t.
3. Substitute t and t' into η and η', respectively.

Figure 3 shows a derivation process in an STSG.

In the rest of the paper, we focus on the rules in which the source elementary tree is not identical to its target, since such identical rules cannot contribute to error correction.

3.2 Inducing an STSG for Error Correction

This section describes a method of inducing an STSG for error correction. The basic idea of our method is similar to the method presented by Dickinson and Meurers (2003b). Their method detects errors by seeking word sequences satisfying the following conditions:

- The word sequence occurs more than once in the corpus.

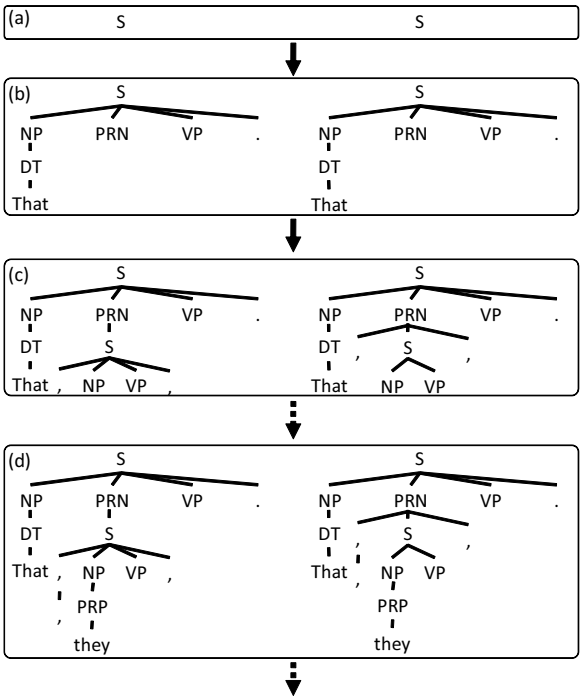

Figure 3: A derivation process of tree pairs in an STSG

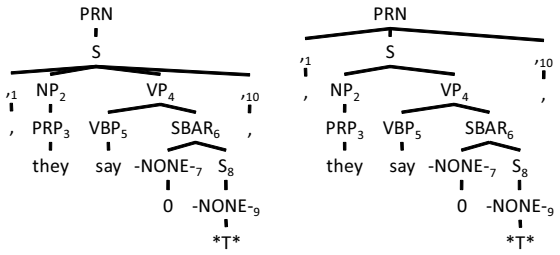

Figure 4: An example of a partial parse tree pair in a pseudo parallel corpus

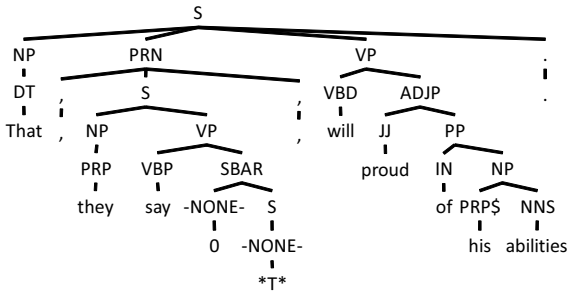

Figure 5: Another example of a parse tree containing a word sequence ", they say ,"

- Different syntactic labels are assigned to the occurrences of the word sequence.

Unlike their method, our method seeks word sequences whose occurrences have different partial parse trees. We call a collection of these word sequences with partial parse trees *pseudo parallel corpus*. Moreover, our method extracts STSG rules which transform the one partial tree into the other.

3.2.1 Constructing a Pseudo Parallel Corpus

Our method firstly constructs a pseudo parallel corpus which represents a correspondence between parse trees containing errors and the ones whose errors are corrected. The procedure is as follows: Let T be the set of the parse trees occurring in the corpus. We write $Sub(\sigma)$ for the set which consists of the partial parse trees included in the parse tree σ. A pseudo parallel corpus $Para(T)$ is constructed as follows:

$$\begin{aligned} Para(T) \;=\; \{\langle \tau, \tau' \rangle \mid \tau, \tau' \in \bigcup_{\sigma \in T} Sub(\sigma) \\ \wedge\, \tau \neq \tau' \\ \wedge\, yield(\tau) = yield(\tau') \\ \wedge\, root(\tau) = root(\tau')\} \end{aligned}$$

where $yield(\tau)$ is the word sequence dominated by τ.

Let us consider an example. If the parse trees depicted in Figure 1 exist in the treebank T, the pair of partial parse trees depicted in Figure 4 is an element of $Para(T)$. We also obtain this pair in the case where there exists not the parse tree (b) depicted in Figure 1 but the parse tree depicted in Figure 5, which contains the word sequence ", they say ,".

3.2.2 Inducing a Grammar from a Pseudo Parallel Corpus

Our method induces an STSG from the pseudo parallel corpus according to the method proposed by Cohn and Lapata (2009). Cohn and Lapata's method can induce an STSG which represents a correspondence in a parallel corpus. Their method firstly determine an alignment of nodes between pairs of trees in the parallel corpus and extracts STSG rules according to the alignments.

For partial parse trees τ and τ', we define a node alignment $C(\tau, \tau')$ as follows:

$$\begin{aligned} C(\tau, \tau') \;=\; \{\langle \eta, \eta' \rangle \mid \eta \in Node(\tau) \\ \wedge\, \eta' \in Node(\tau') \\ \wedge\, \eta \text{ is not the root of } \tau \end{aligned}$$

$\wedge\ \eta'$ is not the root of τ'
$\wedge\ label(\eta) = label(\eta')$
$\wedge\ yield(\eta) = yield(\eta')\}$

where $Node(\tau)$ is the set of the nodes in τ, and $yield(\eta)$ is the word sequence dominated by η. Figure 4 shows an example of a node alignment. The subscripts indicate the alignment.

An STSG rule is extracted by deleting nodes in a partial parse tree pair $\langle \tau, \tau' \rangle \in Para(T)$. The procedure is as follows:

- For each $\langle \eta, \eta' \rangle \in C(\tau, \tau')$, delete the descendants of η and η'.

For example, the rule shown in Figure 2 is extracted from the pair shown in Figure 4.

3.3 Rule Selection

Some rules extracted by the procedure in Section 3.2 are not useful for error correction, since the pseudo parallel corpus contains tree pairs whose source tree is correct or whose target tree is incorrect. The rules which are extracted from such pairs can be harmful. To select rules which are useful for error correction, we define a score function which is based on the occurrence frequencies of elementary trees in the treebank. The score function is defined as follows:

$$Score(\langle t, t' \rangle) = \frac{f(t')}{f(t) + f(t')}$$

where $f(\cdot)$ is the occurrence frequency in the treebank. The score function ranges from 0 to 1. We assume that the occurrence frequency of an elementary tree matching incorrect parse trees is very low. According to this assumption, the score function $Score(\langle t, t' \rangle)$ is high when the source elementary tree t matches incorrect parse trees and the target elementary tree t' matches correct parse trees. Therefore, STSG rules with high scores are regarded to be useful for error correction.

4 An Experiment

To evaluate the effectiveness of our method, we conducted an experiment using the Penn Treebank (Marcus et al., 1993).

We used 49208 sentences in Wall Street Journal sections. We induced STSG rules by applying our method to the corpus. We obtained 8776 rules. We measured the precision of the rules. The precision is defined as follows:

$$precision = \frac{\textit{\# of the positions where an error is corrected}}{\textit{\# of the positions to which some rule is applied}}$$

We manually checked whether each rule application corrected an error, because the corrected treebank does not exist[2]. Furthermore, we only evaluated the first 100 rules which are ordered by the score function described in Section 3.3, since it is time-consuming and expensive to evaluate all of the rules. These 100 rules were applied at 331 positions. The precision of the rules is 71.9%. For each rule, we measured the precision of it. 70 rules achieved 100% precision. These results demonstrate that our method can correct syntactic annotation errors with high precision. Moreover, 30 rules of the 70 rules transformed bracketed structures. This fact shows that the treebank contains structural errors which cannot be dealt with by the previous methods.

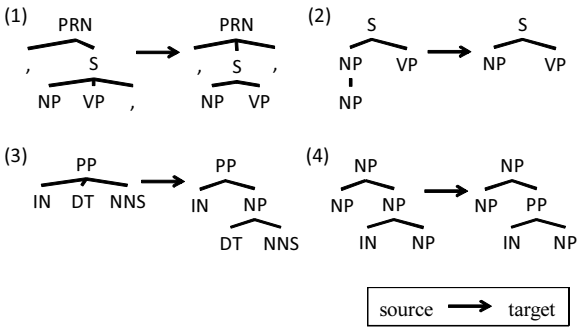

Figure 6: Examples of error correction rules induced from the Penn Treebank

Figure 6 depicts examples of error correction rules which achieved 100% precision. Rule (1), (2) and (3) are rules which transform bracketed structures. Rule (4) simply replaces a node label. Rule (1) corrects an erroneous position of a comma (see Figure 7 (a)). Rule (2) deletes a useless node `NP` in a subject position (see Figure 7 (b)). Rule (3) inserts a node `NP` (see Figure 7 (c)). Rule (4) replaces a node label `NP` with the correct label `PP` (see Figure 7 (d)). These examples demonstrate that our method can correct syntactic annotation errors.

Figure 8 depicts an example where our method detected an annotation error but could not correct it. To correct the error, we need to attach the node

[2]This also means that we cannot measure the recall of the rules.

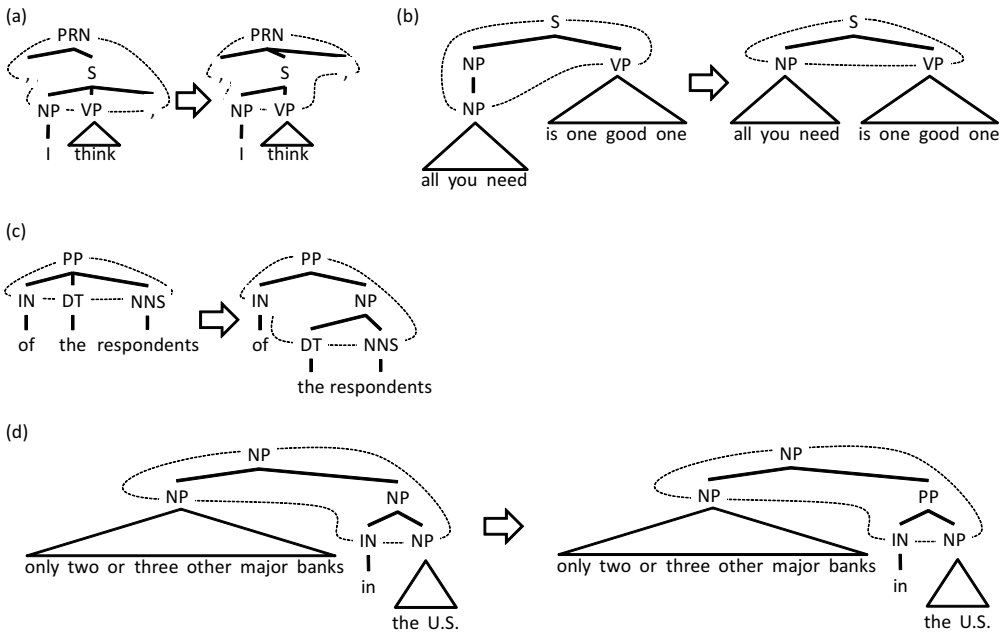

Figure 7: Examples of correcting syntactic annotation errors

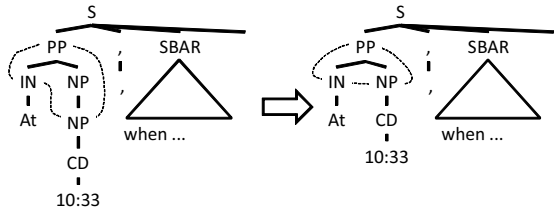

Figure 8: An example where our method detected an annotation error but could not correct it

`SBAR` under the node `NP`. We found that 22 of the rule applications were of this type.

Figure 9 depicts a false positive example where our method mistakenly transformed a correct syntactic structure. The score of the rule is very high, since the source elementary tree `(TOP (NP NP VP .))` is less frequent. This example shows that our method has a risk of changing correct annotations of less frequent syntactic structures.

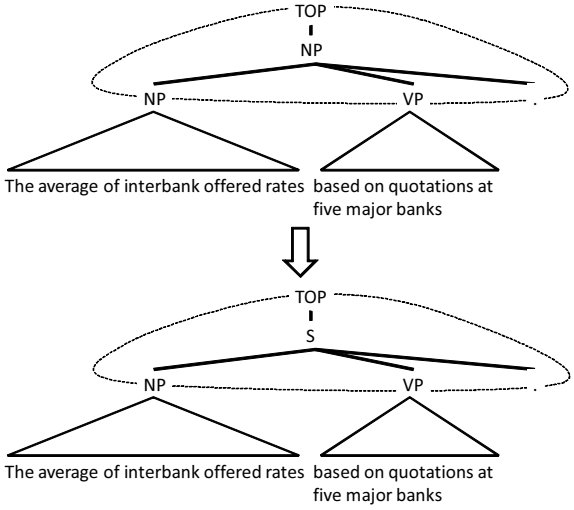

Figure 9: A false positive example where a correct syntactic structure was mistakenly transformed

5 Conclusion

This paper proposes a method of correcting errors in a treebank by using a synchronous tree substitution grammar. Our method constructs a pseudo parallel corpus from the treebank and extracts STSG rules from the parallel corpus. The experimental result demonstrates that we can obtain error correction rules with high precision.

In future work, we will explore a method of increasing the recall of error correction by constructing a wide-coverage STSG.

Acknowledgements

This research is partially supported by the Grant-in-Aid for Scientific Research (B) (No. 22300051) of JSPS and by the Kayamori Foundation of Informational Science Advancement.

References

Adriane Boyd, Markus Dickinson, and Detmar Meurers. 2008. On detecting errors in dependency treebanks. *Research on Language and Computation*, 6(2):113–137.

Trevor Cohn and Mirella Lapata. 2009. Sentence compression as tree transduction. *Journal of Artificial Intelligence Research*, 34(1):637–674.

Markus Dickinson and Detmar Meurers. 2003a. Detecting errors in part-of-speech annotation. In *Proceedings of the 10th Conference of the European Chapter of the Association for Computational Linguistics*, pages 107–114.

Markus Dickinson and Detmar Meurers. 2003b. Detecting inconsistencies in treebanks. In *Proceedings of the Second Workshop on Treebanks and Linguistic Theories*.

Markus Dickinson and W. Detmar Meurers. 2005. Prune diseased branches to get healthy trees! how to find erroneous local trees in a treebank and why it matters. In *Proceedings of the 4th Workshop on Treebanks and Linguistic Theories*.

Jason Eisner. 2003. Learning non-isomorphic tree mappings for machine translation. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics, Companion Volume*, pages 205–208.

Eleazar Eskin. 2000. Detecting errors within a corpus using anomaly detection. In *Proceedings of the 1st North American chapter of the Association for Computational Linguistics Conference*, pages 148–153.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English: the Penn Treebank. *Computational Linguistics*, 19(2):310–330.

Masaki Murata, Masao Utiyama, Kiyotaka Uchimoto, Hitoshi Isahara, and Qing Ma. 2005. Correction of errors in a verb modality corpus for machine translation with a machine-learning method. *ACM Transactions on Asian Language Information Processing*, 4(1):18–37.

Tetsuji Nakagawa and Yuji Matsumoto. 2002. Detecting errors in corpora using support vector machines. In *Proceedings of the 19th Internatinal Conference on Computatinal Linguistics*, pages 709–715.

Tylman Ule and Kiril Simov. 2004. Unexpected productions may well be errors. In *Proceedings of 4th International Conference on Language Resources and Evaluation*, pages 1795–1798.