# Using Tree Transducers for Detecting Errors in a Treebank of Polish

Katarzyna Krasnowska, Witold Kieraś, Marcin Woliński, and Adam Przepiórkowski

Institute of Computer Science, Polish Academy of Sciences, Warsaw, Poland

**Abstract.** The paper presents a modification — aimed at highly inflectional languages — of a recently proposed error detection method for syntactically annotated corpora. The technique described below is based on Synchronous Tree Substitution Grammar (STSG), i.e. a kind of tree transducer grammar. The method involves induction of STSG rules from a treebank and application of their subset meeting a certain criterion to the same resource. Obtained results show that the proposed modification can be successfully used in the task of error detection in a treebank of an inflectional language such as Polish.

## 1 Introduction

Treebanks are an important type of linguistic resource and are currently maintained or developed for numerous languages. Given their crucial role in the task of training probabilistic parsers and, hence, in many natural language processing applications, it is necessary to ensure their high quality. One of the ways to eradicate erroneous structures in a treebank is to develop a method of automated detection of wrongly annotated structures once the resource is created. In this work, we decribe such a method of finding errors in a syntactically annotated corpus and present the results of using it on a treebank of Polish. The paper is divided into two parts: § 2 introduces the method of error detection, and § 3 describes the experiment conducted on the treebank and its results.

## 2 An STSG-Based Approach to Error Detection

### 2.1 Synchronous Tree Substitution Grammars

A Synchronous Tree Substitution Grammar [1] is a set of rules which can be seen as a tree transducer. Each rule comprises of a pair of elementary trees $\langle \tau_1, \tau_2 \rangle$ and a one-to-one alignment between their frontier (leaf) nodes. $\tau_1$ and $\tau_2$ are called *source* and *target*. The derivation starts with two tree roots, left and right (initially with no children), and results in a pair of complete trees. Rule application substitutes source and target into aligned nodes in the left and the right tree, respectively. In order to allow a substitution to be performed, the labels of the substitution node and the substituted tree's root must be identical.

Given a treebank with annotation and structural errors, a set of STSG rules can be induced from it and then used for error detection and correction. The following subsection describes the basic methods of obtaining an STSG grammar introduced by Cohn and Lapata [2] and its application for treebank correction according to Kato and Matsubara [3]. Proposed modifications and extensions are explained in the subsequent subsection.

## 2.2 The Basic Procedure

When inducing an STSG from a syntactically annotated corpus, a *pseudo parallel corpus* is first created [2]. Let $T$ be the set of all syntactic trees in the treebank. The pseudo parallel corpus $Para(T)$ is then the set of all pairs of subtrees found in $T$ such that the two trees in the pair have the same root label and yield (i.e. the terminal sequence dominated by the root), but differ in structure. Formally, $Para(T)$ is given by:

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

where $Sub(\sigma)$ is the set of all subtrees of $\sigma$.

From each tree pair $\langle \tau_1, \tau_2 \rangle \in Para(T)$, an STSG rule is extracted as follows. First, an alignment $C(\tau_1, \tau_2)$ between the nodes of the two trees is determined:

$$\begin{aligned} C(\tau, \tau') = \{\langle \eta, \eta' \rangle \quad | \quad & \eta \in Nodes(\tau) \wedge \eta' \in Nodes(\tau') \\ & \wedge \eta \neq root(\tau) \wedge \eta' \neq root(\tau') \\ & \wedge label(\eta) = label(\eta') \\ & \wedge yield(\eta) = yield(\eta')\}. \end{aligned}$$

Then, for each node pair $\langle \eta_1, \eta_2 \rangle \in C(\tau_1, \tau_2)$, the descendants of both $\eta_1$ and $\eta_2$ are deleted. If $\tau_1$ and $\tau_2$ still differ after this operation, the rule $\langle \tau_1, \tau_2 \rangle$ is added to the grammar.

Rules extracted this way must be filtered so as to eliminate the ones which would create errors instead of correcting them. For this purpose, we follow Kato and Matsubara [3] in using a *Score* function predicting the soundness of a rule. Given a rule $\langle \tau_1, \tau_2 \rangle$, its *Score* is calculated as:

$$Score(\langle \tau_1, \tau_2 \rangle) = \frac{f(\tau_1)}{f(\tau_1) + f(\tau_2)},$$

where $f(\tau)$ is the frequency of $\tau$ in the corpus. Only rules whose *Score* is not lower than a fixed threshold are taken into account.

The resulting set of rules represents structures which are probably erroneous (sources of rules) and their correct counterparts (targets).

## 2.3 Adaptation to a Treebank of Polish

The treebank we work with is Składnica [4]. It is a bank of constituency trees for Polish currently consisting of about 8,200 trees but still under development. The treebank is being developed in a semi-automatic manner. Sets of candidate parse trees are generated by a parser and subsequently one tree is selected by human annotators. This procedure probably leads to a different type of inconsistencies than in a treebank built fully manually. The structures annotators can choose are limited by the grammar, which should make the trees more uniform than those created manually. On the other hand, when a complete tree is presented to the annotator, it is a considerable piece of information, so he or she can easily overlook problems in the details.

Unlike Kato and Matsubara [3], when constructing the pseudo parallel corpus, we compared the yield of the trees in terms of morphosyntactic tags rather than orthographic forms. There were several reasons for choosing such an approach. First, the treebank is currently relatively small, therefore an insufficient number of sentences with a common subsequence of words can be found in it to produce relevant STSG rules. Moreover, Polish inflection makes it even less probable to have two strictly identical word sequences in a corpus (see Figure 1 for an example). The sparseness of the pseudo parallel corpus resulting from these problems can be overcome by abstracting from the orthographic and base forms of the words, and using morphosyntactic tags instead. This way we gain more material for STSG rule extraction since we are capable to draw a parallel between analogous phrases which use different words (e.g. *very small house* vs. *rather thick book*).

As an exception to this decision, particles are not replaced by their tags, but, instead, represented by their base forms (which often are their only possible forms). This was motivated by the fact that particles play important and at the same time very different role in the structure of sentences; compare, e.g., the negative marker *nie* and the subjunctive marker *by*, both analysed as particles.

Another feature of Składnica which requires special treatment is the fact that nonterminal nodes contain not only syntactic categories, but also morphosyntactic features such as gender, number or case. These must be taken into account while extracting and applying STSG rules in order to reflect modifier node agreement and avoid overgeneralisation of rules. A selected subset of feature values of nonterminal nodes is therefore preserved in the pseudo parallel corpus and in the extracted rules.

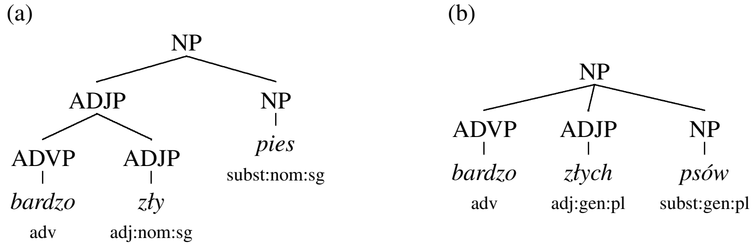

**Fig. 1.** Two syntactic trees for the *NP "very bad dog"* (a) in the nominative singular and (b) in the genitive plural. Even though these sentences clearly represent inconsistent bracketings, no rule would be derived from such trees if orthographic forms of lexemes were taken into consideration.

On the other hand, if rules were kept in such a detailed form, two problems would appear. First, a rule, e.g., for noun phrases in the accusative case, would not be matched by a linguistically relevant structure isomorphic with its source, but occurring, say, in the instrumental. What is more, redundant rules would be created for each combination of morphosyntactic features. As a solution, once a rule is extracted, its feature values are substituted with variables, but retaining any information about agreement, by using the same variable where necessary. Figure 2 shows one example of such an extracted STSG rule, with particular values of number, case, gender and person replaced by variables.

## 3 Experiments on Składnica

A set of 38 rules with *Score* 0.5 or higher was extracted from the treebank. 323 structures matching the source of some rule were found in 302 trees (in 283 trees one rule was matched, in 17 trees — 2 rules, and in 2 trees — 3 rules). Constructions recognised as errors, as well as proposed corrections (i.e. targets of the matched rules), were manually examined and classified into 5 categories presented and explained in Table 1.

Classification into *ERR* and *INC* is motivated by the question of how many of the structures found by the algorithm are syntactically incorrect, thus belonging to the *ERR* class, and how many represent theoretically possible bracketings which, however, are not compliant with the annotation conventions of the treebank (e.g. binary trees for multiple-modifier NPs detected by the rule in Figure 2); in the latter case the class is *INC*. 185 structures in total were assigned to classes other than *FP*, which means that they were wrongly annotated.

Taking into account the classification introduced in Table 1, we propose the following measures of precision:

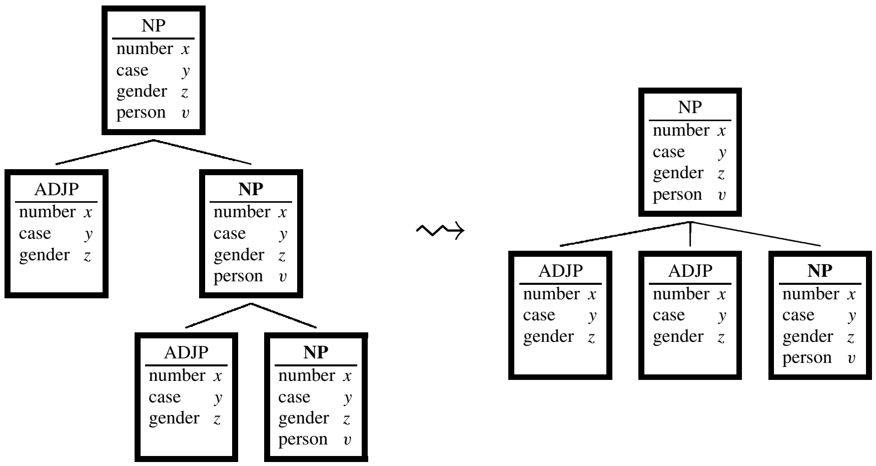

**Fig. 2.** An example rule flattening the structure of an NP modified by two ADJPs. The source tree is inconsistent with annotation guidelines. The rule matches if the ADJPs agree with the NP in number, case and gender.

**Table 1.** Classification of structures found using the STSG rules

| Category | Occurrences | | Comment |
|---|---|---|---|
| $ERR_0$ | 74 | 22.9% | An error in annotation was found and the proposed modification was correct. |
| $INC_0$ | 89 | 27.6% | An inconsistency in annotation was found and the proposed modification was correct. |
| $ERR_1$ | 18 | 5.6% | An error in annotation was found, but the proposed modification was incorrect. |
| $INC_1$ | 4 | 1.2% | An inconsistency in annotation was found, but the proposed modification was incorrect. |
| $FP$ | 134 | 42.7% | A false positive — correct structure pointed out as erroneous. |

$$P_0 = \frac{ERR_0 + INC_0}{ALL},$$

$$P_1 = \frac{ERR_0 + INC_0 + ERR_1 + INC_1}{ALL},$$

$$P_{err} = \frac{ERR_0 + ERR_1}{ALL},$$

where *ALL* stands for all structures retrieved by the rules. The $P_0$ measure is more restrictive and only accepts correct modifications, whereas $P_1$ takes into consideration all correctly recognized wrong structures.

As far as recall is concerned, it is difficult to estimate it without manually inspecting the whole treebank for errors and inconsistencies. In the case of Składnica, such an experiment was performed by its authors [4] on a random sample of 100 sentences and 18 trees were considered wrong. We therefore assume, for the sake of estimating recall, that the whole treebank contains ∼18% (1366) erroneous or inconsistent structures.

Table 2 shows the $P_0$ and $P_1$ precisions, as well as the estimated recall ($R_\sim$) and $F$-measure values. For calculating $R_\sim$ and $F$-measure, all correctly detected wrong trees were treated as true positives (as in $P_1$). The measures were calculated for the results of applying all 38 rules mentioned above (i.e. *Score* threshold 0.5), as well as only those with *Score* equal or above 0.6, 0.7, 0.8 and 0.9. The results show that $P_1$ can be increased by raising the threshold, with the obvious trade-off in $R_\sim$. The highest $F$-measure was achieved for the 0.6 threshold ($R_\sim$ was the same as for 0.5 threshold, i.e. 13.54%, with a slight increase in $P_1$: 57.63% as compared to 57.28%). Increasing the threshold to 0.9 yielded a very high $P_1$ precision of 86.62% at the expense of relatively poor $R_\sim$ and $F$-measure.

22 rules (58% out of 38) achieved a 100% $P_1$ score, and 18 (47%) — a 100% $P_0$ score. The score was lower than 50% for 10 rules in case of $P_1$ and for 12 rules in case of $P_0$. The average rule's $P_1$ and $P_0$ precision were 70.4% and 59.2%, respectively. The plot in Figure 3 shows per-rule $P_1$ and $P_0$ results. Figure 4 presents

**Table 2.** Precision, recall and $F$-measure for different *Score* thresholds

| *Score* threshold | $P_0$ | $P_1$ | $R_\sim$ | $F$-measure |
|---|---|---|---|---|
| 0.5 | 50.46% | 57.28% | 13.54% | 21.91% |
| **0.60** | **50.78%** | **57.63%** | **13.54%** | **21.93%** |
| 0.7 | 50.63% | 57.59% | 13.32% | 21.64% |
| 0.8 | 51.95% | 59.09% | 13.32% | 21.74% |
| 0.9 | 78.17% | 86.62% | 9.00% | 16.31% |

a comparison between $P_1$ and $P_{err}$ precision (i.e. whether a rule detected more errors or inconsistencies). Rule ordering is the same in both plots.

## 4 Related Work

Finding errors in manually annotated corpora is a task actively pursued for well over a decade. Initial research was concerned with finding errors at the level of morphosyntactic annotation, e.g. [5,6,7]; while the methodology of these three works differs considerably, they are all based on the same underlying idea: if similar (in some well-defined way) inputs receive different annotations, the less frequent of these annotations are suspected of being erroneous.

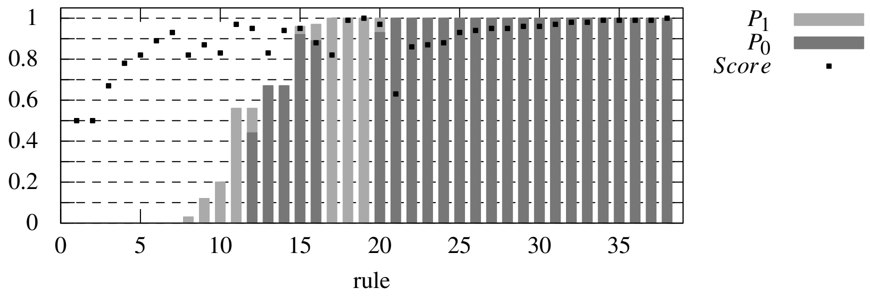

**Fig. 3.** Per-rule precisions $P_1$ and $P_0$

Soon, some of the methods proposed for morphosyntax were generalised to the task of verifying syntactic annotations. In this context, an important line of work is research by Dickinson and Meurers [8,9,10,11]. Before turning to the STSG-based approach described in this work, we had performed some preliminary experiments based on two methods of error detection they proposed: variation $n$-grams [7] and immediate dominance sets [9]. These did not, however, yield promising results, perhaps because of data sparsity (the combination of relatively small treebank and relatively rich morphology); in fact, both precision and recall were very low.[1]

[1] Obviously, these early experiments do not amount to any real evaluation of the methods in question. Rather, the aim of such preliminary experiments was to indicate which method seems to give the quickest returns.

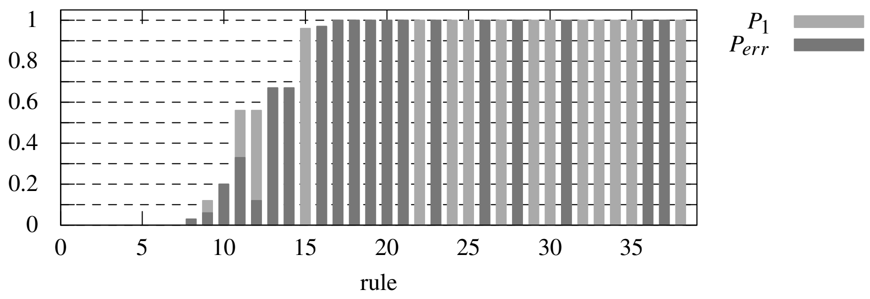

**Fig. 4.** Per-rule precisions $P_1$ and $P_{err}$

## 5 Conclusion

A method of error detection based on the Synchronous Tree Substitution Grammar formalism was implemented and tested on a treebank of Polish. The system found 185 wrongly annotated structures, achieving a precision of 57.63%. As compared to the work of Kato and Matsubara [3], who achieved a precision of 71.9% with their system, corresponding to a total of 331 structures as erroneous, the present result is worse. Nevertheless, it is worth mentioning that they worked on the Wall Street Journal sections of Penn Treebank, which is not only a bigger, but also was created for English, which has a much simpler inflection. We therefore believe that the precision of the current system is relatively satisfying.

As far as recall is concerned, we do not know of any previous results that ours could be compared to, but it is clear that there is still room for improvement. This could be achieved by finding a way to further generalise the extracted rules without loss of information crucial to retaining reasonable precision. For instance, replacing morphosyntactic features in word-level tags with variables, extending the special treatment of particles to other non-inflected parts of speech, or using base forms during the construction of the pseudo parallel corpus (when searching for subtrees with identical yield) all seem worth an experiment.

Although the method presented above gives relatively satisfying results, we are aware that it recognises only certain classes of inconsistencies. In the current treebank, there exist some serial annotation errors that cannot be found using this technique; wrong classification of verbal dependents as arguments or adjuncts is a typical example. The relatively small size of Składnica is also a problem since there is not enough training data to extract rules that can report some less frequent errors. Hence, the need for further research is clear.

**Acknowledgements.** The work described in this paper is partially supported by the DG INFSO of the European Commission through the ICT Policy Support Programme, Grant agreement no.: 271022, as well as by the POIG.01.01.02-14-013/09 project co-financed by the European Union under the European Regional Development Fund.

## References

1. Eisner, J.: Learning non-isomorphic tree mappings for machine translation. In: Proceedings of the 41st Annual Meeting on Association for Computational Linguistics, ACL 2003, vol. 2, pp. 205–208. Association for Computational Linguistics, Stroudsburg (2003)
2. Cohn, T., Lapata, M.: Sentence compression as tree transduction. Journal of Artificial Intelligence Research 34, 637–674 (2009)
3. Kato, Y., Matsubara, S.: Correcting errors in a treebank based on synchronous tree substitution grammar. In: Proceedings of the ACL 2010 Conference Short Papers, ACLShort 2010, pp. 74–79. Association for Computational Linguistics, Stroudsburg (2010)
4. Woliński, M., Głowińska, K., Świdziński, M.: A preliminary version of Składnica — a treebank of Polish. In: Vetulani, Z. (ed.) Proceedings of the 5th Language & Technology Conference, Poznań, pp. 299–303 (2011)
5. van Halteren, H.: The detection of inconsistency in manually tagged text. In: Proceedings of the 2nd Workshop on Linguistically Interpreted Corpora (LINC 2000) (2000)
6. Eskin, E.: Automatic corpus correction with anomaly detection. In: Proceedings of the 1st Meeting of the North American Chapter of the Association for Computational Linguistics (NAACL 2000), Seattle, WA, pp. 148–153 (2000)
7. Dickinson, M., Meurers, W.D.: Detecting errors in part-of-speech annotation. In: Proceedings of the 10nth Conference of the European Chapter of the Association for Computational Linguistics (EACL 2003), Budapest, pp. 107–114 (2003)
8. Dickinson, M., Meurers, W.D.: Detecting inconsistencies in treebanks. In: Nivre, J., Hinrichs, E. (eds.) Proceedings of the Second Workshop on Treebanks and Linguistic Theories (TLT 2003), Växjö, Norway, pp. 45–56 (2003)
9. Dickinson, M., Meurers, W.D.: Prune diseased branches to get healthy trees! How to find erroneous local trees in a treebank and why it matters. In: Civit, M., Kübler, S., Martí, M.A. (eds.) Proceedings of the Fourth Workshop on Treebanks and Linguistic Theories (TLT 2005), Barcelona, pp. 41–52 (2005)
10. Boyd, A., Dickinson, M., Meurers, D.: On detecting errors in dependency treebanks. Research on Language and Computation 6, 113–137 (2008)
11. Dickinson, M., Lee, C.M.: Detecting errors in semantic annotation. In: Proceedings of the Sixth International Conference on Language Resources and Evaluation, LREC 2008, Marrakech, ELRA (2008)