# Detecting syntactic errors in dependency treebanks for morphosyntactically rich languages

Katarzyna Krasnowska and Adam Przepiórkowski

Institute of Computer Science, Polish Academy of Sciences,
`k.krasnowska@phd.ipipan.waw.pl, adamp@ipipan.waw.pl`

**Abstract.** The paper introduces a new method for detecting and correcting errors in large dependency treebanks with rich morphosyntactic annotation. The technique uses error correction rules automatically extracted from the treebank. The procedure of rule extraction is based on a comparison of similar – but not identical – subgraphs of dependency structures. The outcome of applying the method to a 3-million-sentence dependency treebank of Polish is presented and evaluated. The method achieves satisfactory precision in the task of automatic error correction and relatively high precision in the task of error detection.

**Keywords:** dependency treebank, error mining, automatic error detection, automatic error correction

## 1 Introduction

Treebanks are an important type of linguistic resource and are currently maintained or developed for numerous languages. They play a crucial role in the task of training probabilistic parsers and, hence, in many natural language processing applications. This is why it is necessary to ensure their high quality. One of the ways of eradicating erroneous structures in a treebank is to develop a method of automated detection of wrongly annotated structures once the resource is created.

The aim of this paper is to present one such method for detecting errors in a dependency treebank. There were previous reports on application of some methods for pointing out wrongly annotated structures in this type of resource [2, 3]. This paper presents an alternative method, inspired by a technique designed for finding errors in constituency treebanks [5] which was successfully adapted for use with the Polish constituency treebank [6].

The treebank used for the evaluation of the proposed method is an automatically created corpus comprising of a little more than 3 million trees.[1] The method used for creating the treebank (described in detail in [10]) involved the use of a large English-Polish word-aligned parallel corpus. The English part of

[1] The exact number is 3 162 800.

the corpus was parsed automatically using a comprehensive LFG parser for English.[2] Dependency structures for the Polish part were then induced on the basis of the English parse and the word alignment between parallel sentences.

The paper is organised as follows. Section 2 introduces the proposed method of error detection, section 3 describes relevant experiments, and section 4 contains evaluation of the obtained results.

## 2 Method

The proposed method relies on the assumption that constructions which appear in the treebank relatively rarely are likely to be erroneous. Moreover, similar constructions encountered more frequently can be expected to be correct counterparts of the erroneous constructions (the notion of similarity will be explained later on). Although this may not hold in all cases, we are hoping to be able to detect dependency annotation errors with sufficient precision. The idea behind the technique proposed in this paper is thus to define sub-structures of dependency trees that the method will compare, as well as what it means for them to be similar. After extracting relevant structures from the treebank and determining their frequencies, pairings of similar structures are found, possibly representing erroneous constructions and their correct counterparts.

The method is based on connected subgraphs of the dependency trees. As a first step, all such subgraphs of a given size[3] are extracted from each tree. For convenience, we assume that the dependency relations are marked in the child node, not attached to the edge between parent and child, and that the (artificial) root note of the tree is not taken into account for the purpose of subgraph extraction. The subgraphs retain information about parent-child relations and are in fact themselves dependency trees, it is therefore relevant to refer to a subgraph's *root*.

Our approach differs from the one proposed by Kato and Matsubara [5], who extracted subtrees from constituency structures and then truncated them by cutting off all children of specific nodes. In other words, each node in the tree substructures they considered had either all its original children removed, or all its original children retained. In the method proposed here, it is possible for a subgraph's node to retain only a subset of its children from the original dependency tree. In this way the technique presented here achieves greater flexibility, especially given the fact that arguments of verbs can be (and frequently are) omitted or freely ordered in Polish.

Experiments on error detection in small, syntactically annotated corpora of highly inflectional languages, reported in [6], suggest that abstraction from exact word forms (i.e., taking into consideration only their part of speech and

[2] See [1] and `http://www2.parc.com/isl/groups/nltt/xle/doc/xle_toc.html`.

[3] In this work, we only considered subgraphs of sizes 4 and 5 (i.e., based on 4 or 5 words). Smaller subtrees seemed to carry too little information to prove substantially useful. What is more, allowing more possible subtree sizes would increase the already long – because of the treebank's size – time required to run the procedure.

morphological tags) can help in obtaining sufficient amount of data for drawing statistical generalisations. In the case of the study described in this paper, this coarse granularity of information does not seem essential, given the very high number of sentences in the treebank. Moreover, some preliminary experiments conducted with a small (about 8000 trees) dependency treebank described in [9] showed that ignoring word forms may lead to unacceptable loss of lexical information.

For instance, when it abstracted away from lexical information, the method failed in case of sentences containing the common verb *mieć* (*to have*), which has an accusative nominal argument that is not passivisable and is therefore labelled as a complement (COMP), not an object (OBJ), in the dependency schema adopted here. On the other hand, for the vast majority of Polish verbs, the accusative argument is actually a passivisable object. As a result, subgraphs with an OBJ dependency relation between a verb and a noun in accusative case were much more frequent in the treebank than similar subgraphs with a COMP relation, and the method wrongly reported many trees with the verb *mieć* and its accusative complement as errors.

To strike a reasonable balance between the need for generalisation and the necessity to retain some lexical information, graph nodes are represented by base forms of words, together with their CPOS tags[4] and morphological case. This way, the current method is capable of drawing a parallel between, e.g., two sentences containg the same verb with the same arguments, but differing in the verb's person and gender.

To illustrate the above considerations, Figure 1 shows an example dependency tree and all its connected subgraphs of size 4. CPOS tags combined with morphological cases will be referred to as CPOS-case tags. For instance, a noun in dative will be assigned the `noun-DAT` CPOS-case tag.

Once the subgraphs are extracted, all subgraph pairs are found such that:

– their roots are identical in terms of dependency relation, word base form and CPOS-case tag;
– the sequences base forms and CPOS-case tags of all nodes (ordered the same way as corresponding words in the sentence) are identical;
– their internal structures (i.e., subgraph shapes and/or dependency relations) diverge.

An example of a rule extracted in this way from the treebank is presented in Figure 2.

For each rule, all trees containing the source of the rule (the first substructure) are marked as possibly incorrect. Trees created by transforming the subgraph matching the source into one matching the target are suggested as correct.

[4] CPOS tags are coarse-grained POS tags, where fine-grained grammatical classes (e.g., various types of adjectives) are grouped into more traditional parts of speech.

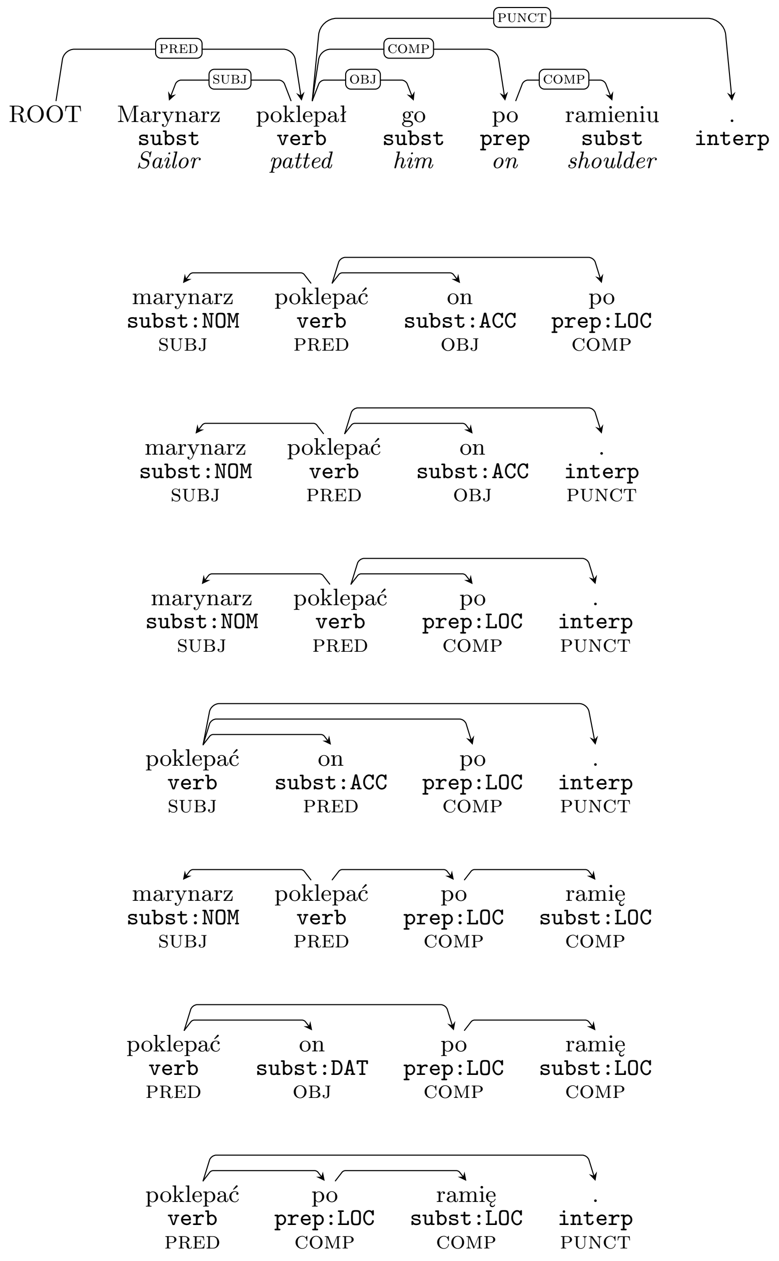

**Fig. 1.** An example dependency tree for the sentence *Marynarz poklepał go po ramieniu.* '*The sailor patted him on the shoulder.*' (taken from the 3-million-sentence dependency treebank) with all its subgraphs of size 4. Orthographic word forms have been replaced in the subgraphs by base forms combined with CPOS-case tags.

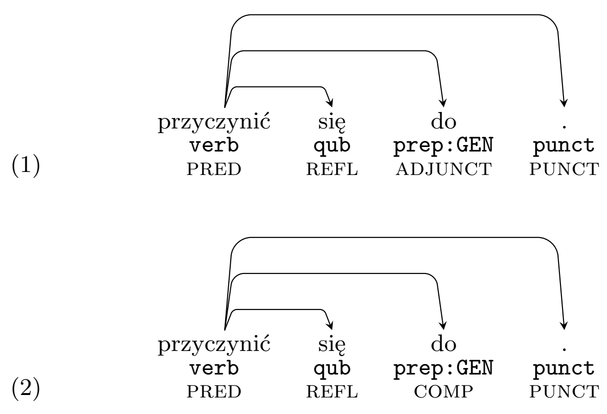

**Fig. 2.** An example rule: (1) is the source of the rule, (2) is the target. The rule changes the dependency relation between the (inherently reflexive) verb *przyczynić się* '*to contribute*' and a preposisional phrase headed by the preposition *do* '*to*' from the incorrect ADJUNCT to the correct COMP.

## 3 Experiments

2 variants of the method described above were run on the 3-million-sentence dependency treebank:

**variant I:** exactly as presented in the previous section;
**variant II:** for some parts of speech, their base forms were ignored, i.e., only CPOS-case tags were taken into account; additionally the dependency relations in the subtrees' roots were ignored.

The modifications to the method were introduced in variant II so as to increase the generality of extracted rules and, as a result, be able to identify more trees as possibly erroneous (see Section 4).

The first modification is motivated by the intuition that for some parts of speech the base form is less important from the point of view of the method. For instance, it is relevant to make a distintion between different verbs because of their different argument structure. On the other hand, it seems that as far as nouns, pronouns, adjectives, numerals and adverbs are concerned, their base form can be omitted without much loss of information, especially, as they typically do not have idiosyncratic combinatorial (or argument structure) properties.

The second introduced modification consists in ignoring dependency relations in the roots of subtrees. Those dependency relations tie the subgraph's root node to its parent from the complete dependency tree. As this parent is not present in the subgraph, this particular dependency relation is much less relevant than the "internal" ones (i.e., those between the subgraph's nodes). What is more, abstracting from the root's dependency relations enables rule extraction in the case where, for example, two similar noun phrases appear as the subject of a sentence and a conjunct in a coordinated structure, respectively.

Table 1 gives the numbers of trees pointed out as wrongly annotated by each variant.

**Table 1.** The number of trees reported as erroneous by the two methods (some trees were reported more than once).

| method variant | I | II |
|---|---|---|
| number of error reports | 18 885 | 852 323 |
| number of distinct trees | 10 237 | 265 460 |
| distinct trees percentage | 54.2% | 31.1% |

## 4 Evaluation

It is difficult to estimate the recall of the implemented method, as the number of erroneous parses in the treebank is not known. It is probably for this reason that some works on error detection do not report recall at all (e.g., [4], [5]). Nevertheless, it is perhaps worthwile to mention two issues concerning recall of error detection. Previous experiments with automatic error detection in a Polish constituency treebank reported in [6] suggest that high recall might be more difficult to achieve than high precision. What is more, the procedure of obtaining the 3-million-sentence dependency treebank used in the experiments involves several steps (sentence alignment, dependency parsing of English sentences, tree projection), all of which are likely to contribute for errors. It is therefore unrealistic to assume that only a small fraction of dependency structures are wrongly annotated (for instance, the percentage of erroneous trees in a Polish constituency treebank, annotated semi-automatically, is estimated to be around 18%, see [8]).

What can be directly estimated is the precision of the method. After applying the method to the treebank, we carefully examined two samples of 100 error reports for each method variant (i.e., four samples were taken into account). Not all error reports could have been included in the samples since preliminary attempts at examining them using the *MaltEval* tool for visualisation[5] showed that some of the structures in the treebank were discontinuous. Since *MaltEval* does not handle discontinuous trees, they were excluded from further examination.

The error reports to include in the samples were chosen as follows. First, an ordered list of rules was created (the orderings were different for each sample type, as explained further on). Second, for each rule, the first tree it indicated as wrong was taken to form a list consisting of one error report per rule. The samples were formed by truncating (i.e., taking only its $n$ first elements for some $n$) the lists so that they contained 100 *distinct* trees. Table 2 gives the size

[5] With the trees after rule application as gold standard, see [7] and `http://w3.msi.vxu.se/~nivre/research/MaltEval.html`.

of sample for each method (sample sizes are greater than 100 since some trees appeared more than once on the list).

**Table 2.** Sample sizes for both variants of the method and both rule ordering strategies. For each sample, the number of trees appearing more than once is also given.

| sample | sample size | trees reported more than once |
|---|---|---|
| $\mathrm{I}_O$ | 107 | 7 |
| $\mathrm{I}_R$ | 103 | 2 |
| $\mathrm{II}_O$ | 118 | 14 |
| $\mathrm{II}_R$ | 101 | 1 |

Two types of sample were created, depending on the rule ordering strategy, as mentioned before. The following ordering strategies were applied:

$O$ — the rules were sorted in the decreasing order of the sum of occurrences of their source and target in the treebank.
$R$ — the rules were sorted randomly.

The first ordering, $O$, is a heuristic for promoting rules which were expected to be more efficient: if there is more material in the treebank to serve as "evidence" for the rule, it might be that not only the rule is more probable to be sound, but also that it detects a common error. The second ordering is expected to allow for better approximation of the method's overall precision. One can think of more possible orderings, e.g., the proportion of rule's source and target occurrences in the treebank (similarly to the approach adopted in [5]).

As a result, four samples were created. The samples will be referred to as $\mathrm{I}_O$ (i.e., method variant I, rule ordering $O$), $\mathrm{I}_R$, $\mathrm{II}_O$, $\mathrm{II}_R$

Each error report from a sample was examined and assigned one of the following categories:

***correct*** for genuine errors with an appropriate correction suggestion,
***partial*** for genuine errors with a wrong correction suggestion,
***wrong*** for correct structures pointed out as erroneous.

In the case of trees which were included in the sample more than once, only one error report, assigned the best category,[6] was taken into account. This is because for each tree, we are interested in whether the method succeeded in detecting an annotation error. In Table 3, the numbers of error reports assigned each category are given.

The precision of each method variant was estimated as the number of reports which pointed out genuine errors divided by the total number of reports

[6] In the sense that *correct* is better than *partial*, and *partial* is better than *wrong*.

**Table 3.** The number of trees for which the error report was assigned each category, given for each evaluated sample.

| sample | *correct* | *partial* | *wrong* |
|---|---|---|---|
| $\mathrm{I}_O$ | 53 | 30 | 17 |
| $\mathrm{I}_R$ | 42 | 30 | 28 |
| $\mathrm{II}_O$ | 57 | 19 | 24 |
| $\mathrm{II}_R$ | 52 | 21 | 27 |

considered, i.e., 100. Two measures of precision were used. The first one, $P_0$, is the number of reports classified as *correct* divided by 100. The second one, $P_1$, was less strict in that it also admitted *partial* error reports. In other words, $P_0$ is the fraction of correctly identified errors with a good correction suggestion, while $P_1$ is the fraction of correctly identified errors regardless of whether their correction suggestion was appropriate. Estimations for $P_0$ and for $P_1$, depending on method variant and rule sorting strategy, are given in Table 4.

**Table 4.** Precision estimates for each sample.

| sample | $P_0$ | $P_1$ |
|---|---|---|
| $\mathrm{I}_O$ | 53% | 83% |
| $\mathrm{I}_R$ | 42% | 72% |
| $\mathrm{II}_O$ | 57% | 76% |
| $\mathrm{II}_R$ | 52% | 73% |

It is clear that the method is much more efficient when it comes to the simple detection of errors, but the precision of error correction is also satisfactory. This makes the proposed method a good candidate for use in semi-automatic error correction, where the correction suggestions are presented to a human annotator who can accept, modify or reject the correction suggested by the system. Variant II of the method outperforms variant I in terms of $P_0$. As far as $P_1$ is concerned, variant II also achieved higher results with exception of the case where $O$ strategy was adopted. Higher precision estimates obtained using the $O$ rule ordering than when the $R$ ordering was applied show that it can be worthwile to somehow arrange the error reports (perhaps using a more sophisticated strategy) in the case where for some reason not all of them can be examined (e.g., due to the large treebank size).

Given the estimation for precision, it is possible to calculate the estimated number of annotation errors that the method managed to find – it can be approximated by the estimated precision multiplied by the total number of distinct trees pointed out as possibly erroneous. As in the case of precision, two estimations can be given depending on whether $P_0$ or $P_1$ is taken into account. As stated before, we are not able to compare them to the actual number of erroneous structures in the treebank, but suspect that many errors are still left undetected given the estimates ranging from 2.1% to 6.4%. Table 5 presents the estimated numbers of correctly identified errors for both method variants.

**Table 5.** The approximate number of all errors found by the method based on precision estimations $P_0$ and $P_1$. The percentages below numbers are the proportions of the approximate number of found errors to the whole treebank size.

| sample | approx. number of errors | |
|---|---|---|
| | using $P_0$ | using $P_1$ |
| $\mathrm{I}_O$ | 82975<br>2.6% | 129943<br>4.1% |
| $\mathrm{I}_R$ | 65754<br>2.1% | 112721<br>3.6% |
| $\mathrm{II}_O$ | 151312<br>4.8% | 201749<br>6.4% |
| $\mathrm{II}_R$ | 138039<br>4.4% | 193785<br>6.1% |

Figures 3 and 4 present two examples of detected errors together with the correct version of the tree suggested by the method as a replacement. For clarity, only the fragment of the tree affected by the rule is shown. In the case of the sentence in Figure 3, the phrases *Commission* and *about this intention* were wrongly annotated as two adjuncts, whereas the correct dependency relations between them and their head *inform* are OBJ (object) and COMP (complement), respectively, as in the structure proposed by the method. In the second case, presented in Figure 4, the phrase *applying sanctions* was assigned a wrong dependency relation and the phrase *from institutions* had a wrong head (*applying* instead of *request*) and a wrong dependency relation. Both errors are corrected in the alternative structure proposed by the method.

## 5 Conclusions

A method for detecting and correcting annotation errors in a dependency treebank for a highly inflectional language was proposed, implemented and evaluated. The evaluation showed that the method achieves reasonable estimated precision for error correction (52%) and good estimated precision when the task is limited to error detection (73%).

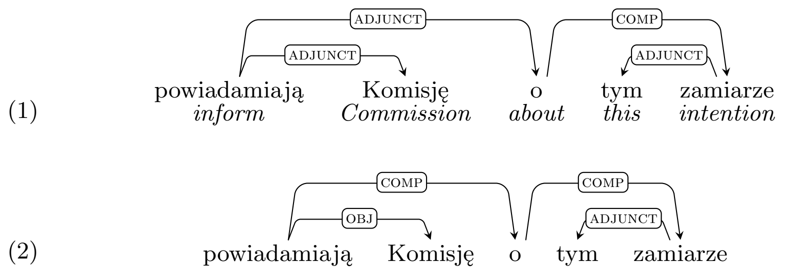

**Fig. 3.** An example of error detected by the method: (1) wrong dependency structure from the treebank (2) proposed correction.

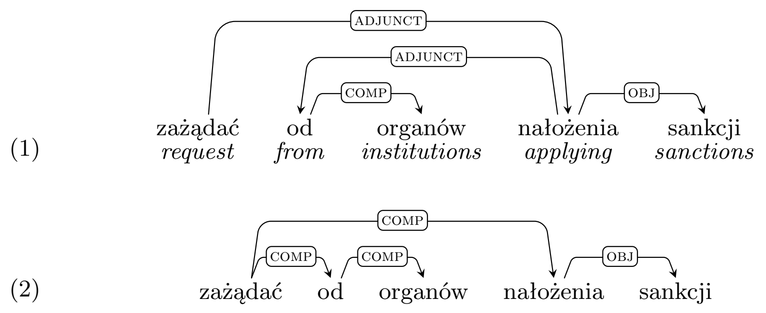

**Fig. 4.** Another example of error detected by the method: (1) wrong dependency structure from the treebank (2) proposed correction.

## 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. Miriam Butt, Helge Dyvik, Tracy Holloway King, Hiroshi Masuichi, and Christian Rohrer. The parallel grammar project. In *Proceedings of the COLING 2002 Workshop on Grammar Engineering and Evaluation*, pages 1–7, Taipei, 2002.
2. Markus Dickinson. Correcting dependency annotation errors. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics (EACL-09)*, Athens, Greece, 2009.

3. Markus Dickinson. Detecting errors in automatically-parsed dependency relations. In *The 48th Annual Meeting of the Association for Computational Linguistics (ACL-10)*, Uppsala, Sweden, 2010.
4. Markus Dickinson and W. Detmar Meurers. Detecting inconsistencies in treebanks. In Joakim Nivre and Erhard Hinrichs, editors, *Proceedings of the Second Workshop on Treebanks and Linguistic Theories (TLT 2003)*, pages 45–56, Växjö, Norway, 2003.
5. Yoshihide Kato and Shigeki Matsubara. Correcting errors in a treebank based on synchronous tree substitution grammar. In *Proceedings of the ACL 2010 Conference Short Papers*, ACLShort '10, pages 74–79, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics.
6. Katarzyna Krasnowska, Witold Kieraś, Marcin Woliński, and Adam Przepiórkowski. Using tree transducers for detecting errors in a treebank of Polish. In Petr Sojka, Aleš Horák, Ivan Kopeček, and Karel Pala, editors, *Text, Speech and Dialogue: 15th International Conference, TSD 2012, Brno, Czech Republic*, volume 7499 of *Lecture Notes in Artificial Intelligence*, pages 119–126. Springer-Verlag, Heidelberg, 2012.
7. Jens Nilsson and Joakim Nivre. MaltEval: An evaluation and visualization tool for dependency parsing. In *In Proceedings of the Sixth International Language Resources and Evaluation. LREC*, 2008.
8. Marcin Woliński, Katarzyna Głowińska, and Marek Świdziński. A preliminary version of Składnica—a treebank of Polish. In Zygmunt Vetulani, editor, *Proceedings of the 5th Language & Technology Conference*, pages 299–303, Poznań, 2011.
9. Alina Wróblewska. Polish dependency bank. *Linguistic Issues in Language Technology*, 7(1), 2012.
10. Alina Wróblewska and Adam Przepiórkowski. Induction of dependency structures based on weighted projection. In *Proceedings of the 4th International Conference on Computational Collective Intelligence Technologies and Applications (ICCCI 2012), Part I*, volume 7653 of *Lecture Notes in Artificial Intelligence*, pages 364–374, Berlin, 2012. Springer-Verlag.