Parse Me if You Can: Artificial Treebanks for Parsing Experiments on Elliptical Constructions

Kira Droganova,[1] **Daniel Zeman,**[1] **Jenna Kanerva,**[2] **Filip Ginter**[2]
[1]Charles University, Faculty of Mathematics and Physics, Praha, Czechia
[2]TurkuNLP Group, University of Turku, Finland
[1]{droganova, zeman}@ufal.mff.cuni.cz, [2]{jmnybl, figint}@utu.fi

Abstract

In this work we focus on a particular linguistic phenomenon, ellipsis, and explore the latest parsers in order to learn about parsing accuracy and typical errors from the perspective of elliptical constructions. For this purpose we collected and processed outputs of several state-of-the art parsers that took part in the CoNLL 2017 Shared Task. We extended the official shared task evaluation software to obtain focused evaluation of elliptical constructions. Since the studied structures are comparatively rare, and consequently there is not enough data for experimentation, we further describe the creation of a new resource, a semi-artificially constructed treebank of ellipsis.

Keywords: Ellipsis, Syntactic parsing, Evaluation, Universal Dependencies

1. Introduction

Ellipsis, i.e. omission of linguistic content that is silently understood by both the speaker and the addressee, is a phenomenon present—in various forms—in many natural languages. Ellipsis obviously makes natural language understanding harder; but sometimes it also complicates syntactic parsing of the content that is not omitted. In dependency syntax (which is the framework within which we operate), a parent node may be missing while its dependents are present. One might either create an "empty" node for the missing word, or choose a substitute parent among the words that are not missing. Both options make parsing difficult: in the former case, the parser must learn where to generate empty nodes; in the latter, relations are drawn between nodes that would not be connected otherwise, hence they are not easily learned from data.

In any case, modern dependency parsers *are* data-driven and they can hardly account for those types of ellipsis that are not represented in training data. If the data contains empty nodes, the parser can try to learn generating them. If the data does not contain any specific annotation of ellipsis, we have to hope that the parser learns to occasionally attach dependents to strange parents, even without knowing that it is ellipsis what caused the lack of better options.

In this study we focus on elliptical constructions in the so-called *basic representation* of Universal Dependencies (UD) (Nivre et al., 2016). The annotation style of UD does not mark ellipsis explicitly when it does not have to: most types are solved by simply promoting one orphaned dependent to the position of its missing parent. Admittedly, there are treebanks that overtly annotate a wider range of elliptical structures. Our main reason for working with UD is practical: substantial data is available in this annotation style for several dozens of languages, and state-of-the art parsers have been trained and tested on UD.

The one exception where UD explicitly marks ellipsis are certain types of gapping and stripping (Droganova and Zeman, 2017), where multiple orphaned dependents of a missing predicate have to be connected using a special relation called `orphan` (Figure 1). In the present work we investigate how frequent are the `orphan` relations in data, how well can existing parsers learn to recognize them, and how can we extend the data to provide more training material and improve parsing accuracy.

2. Data

For the purpose of the experiments we use the system outputs from the CoNLL 2017 Shared Task (Zeman et al., 2017), that are now available as a corpus. We chose 12 teams whose systems surpassed baseline results (Zeman et al., 2017) on labelled attachment score (LAS): C2L2, darc, HIT-SCIR, IMS, Koç University, LATTICE, NAIST-SATO, Orange-Deskiñ, Stanford, TurkuNLP, ÚFAL-UDPipe 1.2 and UParse.

3. Experiments

The idea behind this work is to look closely at the current parsers regarding their ability to parse non trivial linguistic constructions such as elliptical constructions, and collect the information about typical errors, how they differ from parser to parser.

For the purpose of this experiment we adapted and extended the evaluation script which had been created to evaluate system output files for the 2017 Shared Task. The main idea of such adaptation is to save evaluation techniques that were proposed and implemented by the 2017 task organizers. Since the data was selected relying on these techniques, we hope that following the same line, especially regarding word alignments and sentence segmentation, helps us to be more precise. The script is available at the Shared Task page.[1] The adapted script can be found on GitHub.[2]

The adapted script provides information of two types:

- Statistics on correctly predicted `orphan` relations;
- Statistics on erroneously predicted or missed `orphan` relations and typical errors.

[1] `http://universaldependencies.org/conll17/evaluation.html`

[2] `https://github.com/Kira-D/conll2017/tree/deprelCalc`

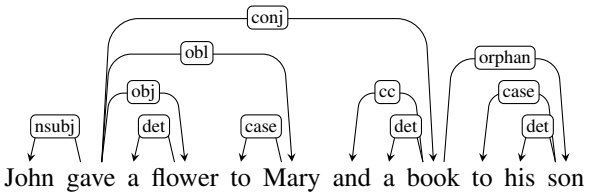

Figure 1: UD v2 uses the orphan relation to attach unpromoted dependents of a predicate to the promoted dependent.

4. Evaluation

Table 1 shows the statistics on correctly predicted orphan relations. In these calculations we use the relative number of all orphan nodes for every team, which is based on alignment between system output words and gold standard words. In other words, only successfully aligned orphan nodes from gold standard are included in this number. It is clearly seen that both Recall and F-measure are rather low. At the same time, percentage of correctly predicted dependency labels for head nodes is quite high.

Table 2 shows the statistics on erroneously predicted or missed orphan labels. For every parser that we selected for the experiment, we calculate error pairs "relation1-relation2", where the first relation was taken from the aligned gold word and the second relation was assigned by the system. Table 2 provides top 5 error pairs. Every cell contains the following information:

- the error pair;
- the contribution of the pair to the number of all errors concerning orphan label (percentage);
- the number of instances of the error type (frequency);
- h.error shows erroneously predicted head nodes (percentage and absolute number).

It seems that parsers make mistakes in similar conditions: the error types and their frequencies are almost the same from parser to parser.

What is important, the number of orphan labels is just a tiny fraction of all labels and the contribution of their low values of Recall and F-measure to the final figures calculated on the whole amount of data goes virtually unseen. Hence, the question is if the parsers perform really poorly on elliptical constructions or it is simply the lack of data. To answer that question, we would need more data; since naturally occurring ellipsis is rare, we decided to artificially create a set of sentences that are structurally similar to natural elliptical constructions.

Parser	All	Correct	Recall	F1	Parent	Parent %
C2L2	1420	217	15.28%	26.48%	192	88.48%
darc	1411	194	13.75%	19.06%	180	92.78%
HIT-SCIR	1411	341	24.17%	34.13%	292	85.63%
IMS	1421	241	16.96%	28.83%	208	86.31%
Koc-University	1420	194	13.66%	20.78%	161	82.99%
LATTICE	1420	200	14.08%	20.62%	166	83.0%
NAIST-SATO	1420	391	27.54%	41.53%	357	91.3%
Orange-Deskin	1420	369	25.99%	35.16%	280	75.88%
Stanford	1420	454	31.97%	49.11%	408	89.87%
TurkuNLP	1420	218	15.35%	23.37%	189	86.7%
UFAL-UDPipe-1-2	1423	226	15.88%	23.69%	182	80.53%
UParse	1420	326	22.96%	33.44%	288	88.34%

Table 1: Correctly predicted orphan relations. Parser: names of the teams in alphabetic order; All: number of orphan labels; Correct: number of correctly predicted orphan labels; Recall: number of correct orphan labels divided by the number of gold-standard orphan nodes; F1: F-measure: 2PR / (P+R); Parent: number of correctly predicted parent nodes; Parent %: percent of correctly predicted parent nodes;

5. Creating Artificial Treebanks

Recent research (Schuster et al., 2017; Droganova and Zeman, 2017) provides a detailed overview of elliptical constructions within the UD framework and presents typical patterns that can be used for detection of elliptic constructions. This information allows us to develop a script that transforms non-elliptic UD style trees to elliptic trees.

Figure 2 shows a subtree pattern that matches sentences where gapping (Johnson, 2009) could potentially occur (but it did not, or at least it was not annotated following the UD guidelines, because there is no orphan relation). An example of an English sentence that matches the pattern: "But not always do those three agree, and not always are their decisions equal."

Figure 3 provides the tree structure of this sentence. It matches the pattern because 1. its "root" node is a verb; 2. the verb has an "aux" child; 3. the verb is linked with another clause via a "conj" relation; 4. the other clause is headed by an adjective and has a "cop" (copula) dependent; 5. both clauses contain a "nsubj" (subject) and an "advmod" (adverbial modifier). After transformation the sentence would lose an adjective and its dependent. The new structure is shown in Figure 4.

It should be mentioned that patterns do not require a particular word order, only particular dependents. Thus the sentence in Figure 5 is a match as well.

The methodology requires manual efforts. After application of the script, the data have to be checked and corrected:

- After artificial omission sentences must remain grammatically correct (Figure 10, Figure 11);
- The patterns are designed to match as many instances as possible, so the erroneous instances have to be filtered out or manually corrected. All sentences at Figures 3, 5 and 10 match the pattern at Figure 2, but after conversion sentence at the Figure 11 becomes ungrammatical. However, it can be fixed manually (Figure 12).

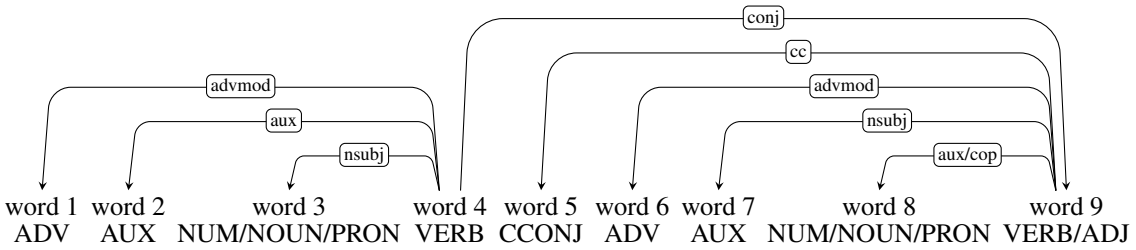

Figure 2: An example of a gapping pattern. Unless explicitly said otherwise, patterns are not word-order-sensitive.

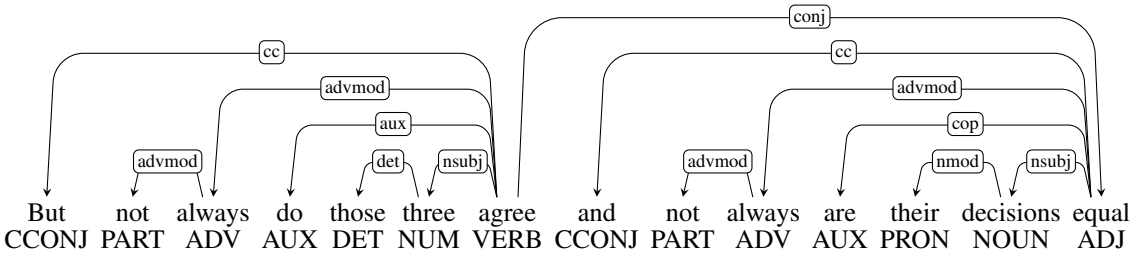

Figure 3: An example of a matched sentence (before conversion).

5.1. Input Data

Our methodology can in principle be applied to any UD treebank. The dataset and experiments presented in this paper are based on Czech, English and Finnish treebanks from UD 2.1 (Nivre et al., 2017).

In addition, large web corpora of the three languages (Zeman et al., 2017) (Ginter et al., 2017) were parsed by two parsers (Stanford (Dozat et al., 2017) and Baseline UDPipe (Zeman et al., 2017) entries in the CoNLL17 Shared Task on Multilingual Parsing from Raw Text to Universal Dependencies) and used as an additional source of trees to get more candidate material. After double parsing, only trees with identical analysis were kept to ensure the quality of the automatic parses. An informal manual inspection of these trees confirmed that the quality is sufficient. All input data are in the CoNLL-U format.[3]

For English, there are four UD treebanks: "Original", LinES, ParTUT, PUD. For Czech, we used the training parts of "Original", CAC and FicTree. For Finnish, there are three treebanks in UD but only the "Original" UD_Finnish was used in the present work, and only for development of the rules, which were then applied to the parsed web corpus to create the final artificial dataset.

5.2. Conversion

The conversion was performed using Udapi, an open-source framework that provides an application programming interface for processing Universal Dependencies data (Popel et al., 2017). Udapi components that were developed for the conversion are not integrated into the framework and cannot be obtained from the the official Udapi project repository due to their extremely narrow specialization. The components can be found on GitHub.[4]

The algorithm consists of three steps. First, we use existing Udapi functionality to filter out sentences that cannot be converted into artificial sentences in any way. For this purpose we composed a filtering query that restricts the structure of a candidate sentence. For instance, gapping occurs in coordinate structures and typically the second clause contains a gap (Coppock, 2001), therefore a candidate sentence must contain at least two clauses and the candidate word must be in the second. By candidate word we mean the token that will be deleted or transformed in the next step.

Another example of a filtering rule is the requirement that the heads of both clauses have at least 3 dependents that are neither function words nor punctuation. (That is, they are arguments or adjuncts.)

One of our observations suggests that elided elements can hardly be restored if some other clause is encountered between the two clauses. Therefore we delete such sentences in the first step.

The filtering query simplifies visualization of candidate trees which is helpful for adjustment of the rules.

Second, we use the Udapi components that we developed in order to propose the conversion. The first component analyzes candidate words within a sentence and duplicates the sentence if there are two or more words marked as candidate and not all of them are verbs. Simultaneously, the component leaves only one word marked as candidate for each copy. The verb restriction prevents such sentences from duplication:

(1) Mary won gold, Peter won silver, and Jane won bronze.

This rule was designed for the following purposes:

- If a candidate sentence contains three or more coordinate clauses, we do not want to allow gapping in the second clause if it does not occur in the third clause. Either both must be gapped or none of them.

[3] http://universaldependencies.org/format.html

[4] https://github.com/Kira-D/UDapy_block_artificial

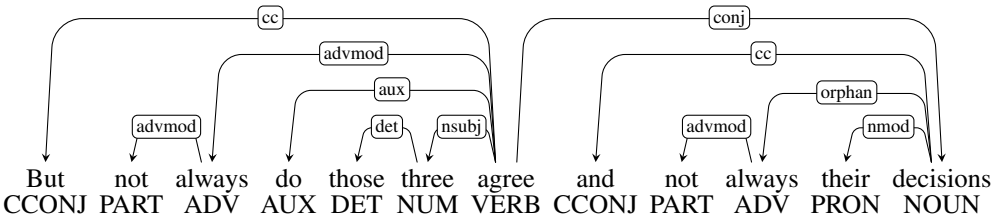

Figure 4: The matched sentence from Figure 3 after conversion.

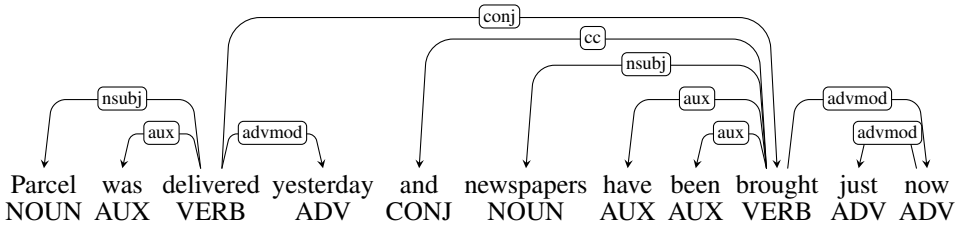

Figure 5: An example of a matched sentence.

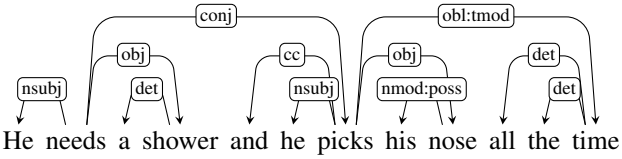

Figure 6: An example of a sentence before Type 1 conversion.

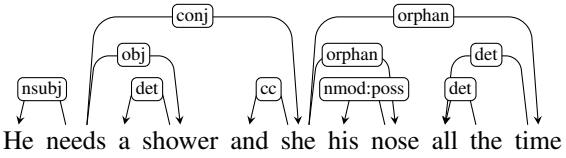

Figure 7: The sentence from Figure 6 after Type 1 conversion.

- If a candidate sentence contains three or more coordinate clauses and some of them are copular clauses, due to the conversion rules the output sentence will contain different types of ellipsis. Consequently, the output structure is not parallel—dependent clauses do not repeat the structure of the main clause. Conversion rules accept copular clauses, but can convert only non-core ellipsis (Figure 8) from such constructions and only if they have enough non-core dependents.

The second component analyzes candidate words and proposes the conversion according to the rules.
The proposed conversion is stored in the CoNLL-U file[5] for every token that should be changed. The information contains either new position in the sentence, parent token and dependency relation, or a marker that the token should be deleted.
The component proposes conversion of two types:

- Type 1 (Core argument ellipsis)—a verb and its auxiliary verbs are marked as 'to be deleted'. Figure 6 and Figure 7 show an example of such conversion.
- Type 2 (Non-core argument ellipsis)—some or even all core arguments are marked as 'to be deleted'. Figure 8 and Figure 9 illustrate this type.

If no rule can be applied to the sentence, the sentence is marked as 'to be deleted'.

[5]The tenth column, MISC, allows for storing arbitrary additional annotation.

Some additional markers were used during the conversion process to adjust the rules and to make sure that the rules cover most of the sentences present in the data.
Third, we use the Udapi component that was developed to perform the final conversion:

- Delete the sentences that are marked as 'to be deleted';
- Delete the tokens that are marked as 'to be deleted';
- Change the sentence structure according to the information that is stored in the MISC column;
- Delete all markers that were added during conversion.

5.3. Complex Issues

We iteratively adjusted the conversion rules, manually checking output samples, making the rules more strict and precise, and re-running them.
However, the output data is still not perfect. We attribute this issue to lack of information: Our rules do not have access to a dictionary and cannot always assess the degree to which the dependents of the two verbs are semantically compatible (Figure 11). Some of the complex sentences may be correctly converted using grammatical cases (see section 5.4.). Some language-specific relation types may help, for instance, English has *obl:tmod* (temporal modifier), a subtype of *obl* (oblique argument or adjunct). Unfortunately, rules relying on relation subtypes are not always applicable to trees produced by parsers because some parsers are trained to only generate the main dependency type—*obl* in this case.

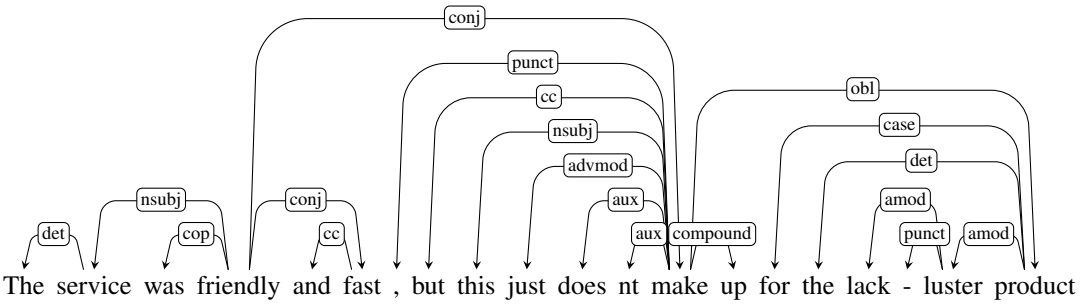

Figure 8: An example of a sentence before Type 2 conversion.

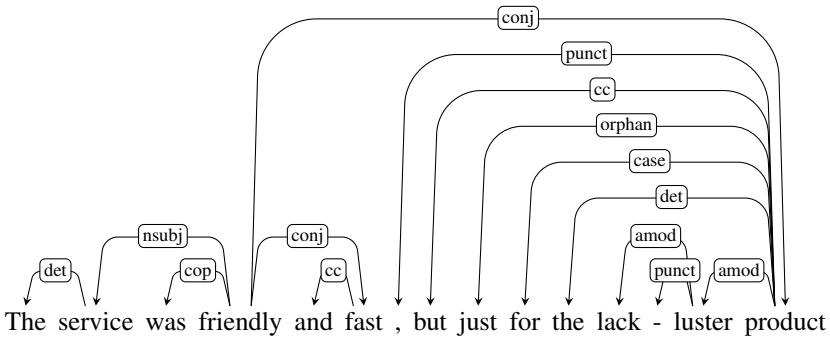

Figure 9: The sentence from Figure 8 after Type 2 conversion.

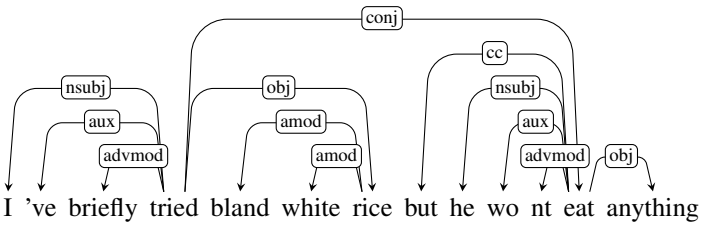

Figure 10: An example of a matched sentence before conversion, problematic case.

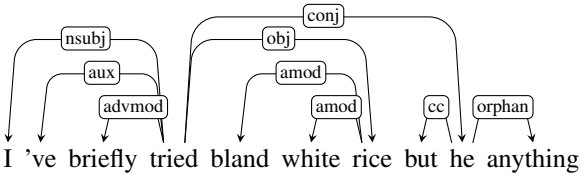

Figure 11: The sentence from Figure 10 after conversion.

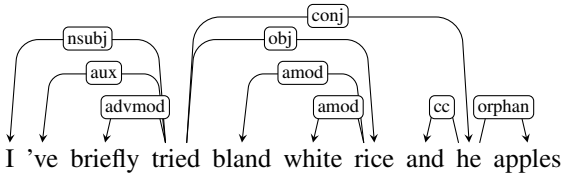

Figure 12: The sentence from Figure 10 after manual correction.

5.4. Language-specific Rules

It is not possible to avoid language-specific rules, although some of them contradict each other. For instance, Finnish allows elliptic sentences with omitted subjects in both clauses and remaining objects (Figure 16, Figure 17), which hardly can be found in English. We thus maintain separate processing pipelines for each language.

An example of an English-specific rule is substitution of personal pronouns. The original sentence typically describes a sequence of actions performed by one actor. By

C2L2	orphan-conj 23.0% 327 h.error: 85.63% 280 orphan-nmod 15.05% 214 h.error: 42.52% 91 orphan-obl 6.12% 87 h.error: 62.07% 54 orphan-advmod 6.05% 86 h.error: 72.09% 62 conj-orphan 5.91% 84 h.error: 61.9% 52
darc	orphan-conj 14.5% 267 h.error: 76.4% 204 orphan-nmod 12.6% 232 h.error: 47.84% 111 conj-orphan 7.98% 147 h.error: 78.23% 115 orphan-obl 5.75% 106 h.error: 55.66% 59 orphan-advmod 5.05% 93 h.error: 60.22% 56
HIT-SCIR	orphan-conj 16.05% 266 h.error: 79.32% 211 orphan-nmod 10.38% 172 h.error: 51.74% 89 conj-orphan 8.99% 149 h.error: 69.13% 103 orphan-obl 6.46% 107 h.error: 68.22% 73 orphan-advmod 4.59% 76 h.error: 80.26% 61
IMS	orphan-conj 22.92% 328 h.error: 81.4% 267 orphan-nmod 14.12% 202 h.error: 43.56% 88 orphan-obl 6.92% 99 h.error: 56.57% 56 orphan-advmod 6.5% 93 h.error: 67.74% 63 conj-orphan 5.87% 84 h.error: 69.05% 58
Koc-University	orphan-conj 20.5% 343 h.error: 79.01% 271 orphan-nmod 12.67% 212 h.error: 52.83% 112 conj-orphan 6.16% 103 h.error: 69.9% 72 orphan-obl 5.5% 92 h.error: 66.3% 61 orphan-advmod 4.72% 79 h.error: 70.89% 56
LATTICE	orphan-conj 17.24% 300 h.error: 82.0% 246 orphan-nmod 13.33% 232 h.error: 49.57% 115 conj-orphan 7.93% 138 h.error: 68.84% 95 orphan-obl 6.03% 105 h.error: 63.81% 67 orphan-advmod 5.11% 89 h.error: 67.42% 60
NAIST-SATO	orphan-conj 17.23% 257 h.error: 82.1% 211 orphan-nmod 11.73% 175 h.error: 45.71% 80 conj-orphan 9.72% 145 h.error: 56.55% 82 orphan-obl 6.7% 100 h.error: 67.0% 67 orphan-advmod 4.83% 72 h.error: 63.89% 46
Orange-Deskin	orphan-conj 15.14% 262 h.error: 71.37% 187 conj-orphan 10.34% 179 h.error: 69.27% 124 orphan-nmod 9.76% 169 h.error: 45.56% 77 orphan-obl 5.6% 97 h.error: 65.98% 64 orphan-advmod 4.68% 81 h.error: 66.67% 54
Stanford	orphan-conj 17.71% 247 h.error: 85.43% 211 orphan-nmod 12.19% 170 h.error: 45.88% 78 conj-orphan 10.9% 152 h.error: 61.84% 94 orphan-obl 5.3% 74 h.error: 64.86% 48 orphan-advmod 5.23% 73 h.error: 65.75% 48
TurkuNLP	orphan-conj 19.96% 329 h.error: 74.77% 246 orphan-nmod 12.38% 204 h.error: 47.55% 97 conj-orphan 8.56% 141 h.error: 73.05% 103 orphan-obl 6.37% 105 h.error: 67.62% 71 orphan-advmod 5.95% 98 h.error: 63.27% 62
UFAL-UDPipe-1-2	orphan-conj 17.12% 288 h.error: 81.94% 236 orphan-nmod 12.31% 207 h.error: 44.93% 93 conj-orphan 8.62% 145 h.error: 73.79% 107 orphan-obl 5.77% 97 h.error: 59.79% 58 orphan-advmod 4.88% 82 h.error: 56.1% 46
UParse	orphan-conj 14.96% 243 h.error: 81.48% 198 orphan-nmod 12.5% 203 h.error: 50.74% 103 conj-orphan 9.11% 148 h.error: 59.46% 88 orphan-obl 5.91% 96 h.error: 69.79% 67 orphan-advmod 4.37% 71 h.error: 59.15% 42

Table 2: Erroneously predicted or missed `orphan` labels and their frequencies

substituting the second subject with a different pronoun, we create an elliptical sentence where two actors perform presumably the same action but with different patients (Figure 6, Figure 7). Another example would be a rule for copular constructions. If the main clause contains a copula, the sentence must be converted into Type 2 structure (Figure 8,

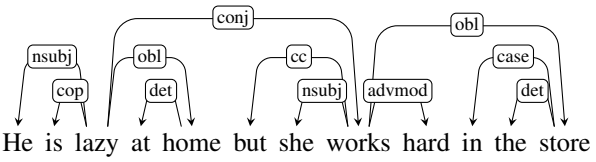

Figure 13: An example of an English sentence with copula before conversion.

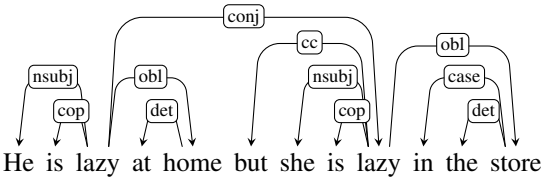

Figure 14: The sentence from Figure 13, copula reconstruction

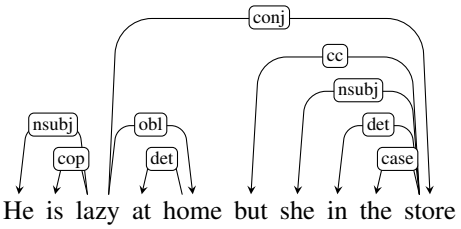

Figure 15: The sentence from Figure 13, correct analysis

Figure 9). Consider an example at Figure 13. After deletion of the verb, the structure would have to be reconstructed as there was a copula (Figure 14). And in this case, if the copula is not repeated, the second clause is interpreted just as a clause with a nonverbal predicate (Figure 15).

In all languages it is important to check that the deleted predicate has at least two arguments or adjuncts whose type (e.g., "nsubj", "obj", "obl:agent") matches dependents of the first predicate; otherwise no gapping can occur. If the dependent is a prepositional phrase, the meaning of the prepositions must be compatible, too (note that it does not necessarily mean that the prepositions are identical). Furthermore, in Czech and Finnish the morphological case of the nouns is important. (English does not have cases, with the exception of personal pronouns.)

Obviously, the rules described above filter out a substantial amount of sentences. Therefore, we currently do not check prepositions in English; instead, we manually fix sentences where prepositions are not compatible.

A good Czech example from the parsed web corpus is in Figures 18 and 19. Both verbs take a directional adjunct with the preposition *do* "to" and a genitive noun, which makes the gapped sentence sound very natural.

6. Results

We provide artificial ellipsis treebanks for three languages, Czech, English and Finnish, using our processing pipeline explained in previous sections. Furthermore, the data for English and Finnish is manually checked and fixed to be grammatical and naturally-sound using UD Annotatrix (Tyers et al., 2017) annotation tool. This further ensures the

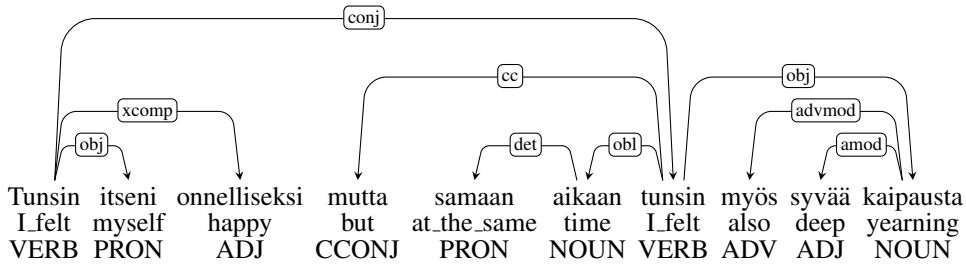

Figure 16: An example of a Finnish sentence automatically identified in the parsed Finnish web corpus.

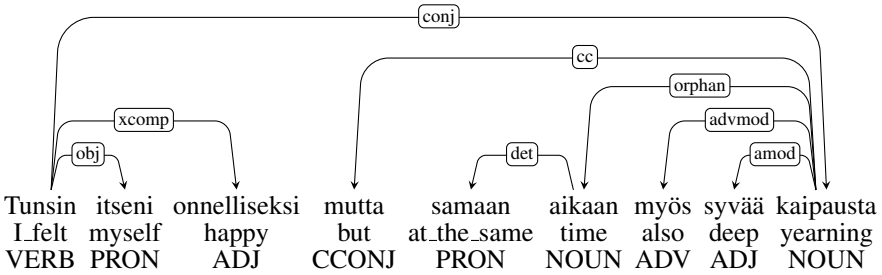

Figure 17: The Finnish sentence from Figure 16 after conversion.

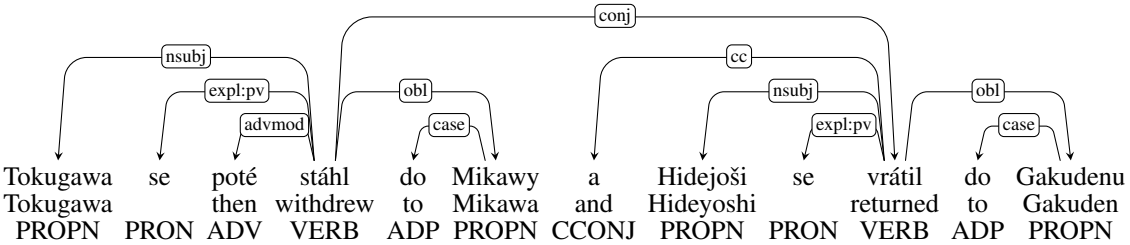

Figure 18: A perfect example automatically identified in the parsed Czech web corpus.

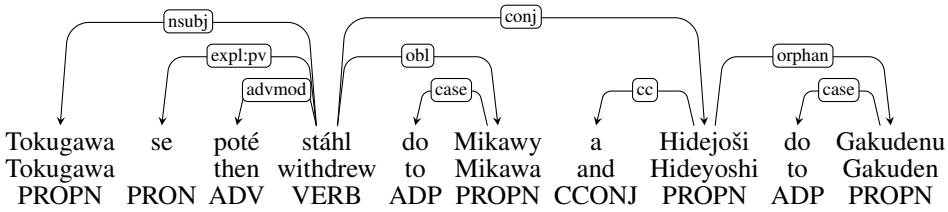

Figure 19: The Czech sentence from Figure 18 after conversion.

good quality of the provided data. Table 3 provides information concerning the sizes of these three datasets.

7. Related Work

The idea of artificial generation or modification of corpora is not new and it has been occasionally applied to various areas of language learning, whenever the studied phenomenon is underrepresented in existing resources. To name just a few: In (van der Plas et al., 2009), creation of an artificial treebank from an existing text treebank helps to overcome domain differences. (Khoshnavataher et al., 2015) artificialy modify text to look like obfuscated plagiarism; the resulting corpus is used to train a plagiarism-detecting system for Persian. And (Gulordava and Merlo, 2016) generate word-order permutations to study the im-

	Initial	Processed	Manual
Czech	1.7M / 102K	13K / 498	NA
Czech web	23M / 2M	37K / 2369	NA
English	408K / 24K	6.8K / 284	3.7K / 183
English web	883K / 89K	6.4K / 422	3.6K / 238
Finnish web	31M / 4.3M	31K / 2442	13K / 1000

Table 3: The size of the data. Initial: the size of the input data, tokens/sentences; Processed: the size of the data after the application of the conversion pipeline, tokens/sentences; Manual: the size of the data after manual correction, tokens/sentences

pact of word order on parsing accuracy in twelve different languages.

8. Conclusion

We have presented experiments that provide a closer look at the current parsers regarding their ability to parse elliptical constructions. We have proposed and described a methodology for creation of artificial treebanks for parsing experiments. The parsing experiments with the artificial data are the natural next step, which we are going to take.

The first version of the artificial elliptic UD treebanks is publicly available in CoNLL-U format via the LINDAT/CLARIN repository at `http://hdl.handle.net/11234/1-2616`.

The artificial data provides a decent amount of ellipsis-like structures. The number of sentences may not look large but it is still several times bigger than naturally occurring elliptical sentences in large treebanks. It will significantly increase the basis that can be used to train ellipsis-aware parsers.

9. Acknowledgements

The work was partially supported by the grant 15-10472S of the Czech Science Foundation (GAČR), by the GA UK grant 794417, Academy of Finland, and Nokia Foundation. The data is available via the LINDAT/CLARIN repository, which is part of a research infrastructure project funded by the Ministry of Education, Youth and Sports of the Czech Republic, Project No. LM2015071. Computational resources were provided by CSC - IT Center for Science, Finland.

10. Bibliographical References

Coppock, E. (2001). Gapping: In defense of deletion. In *Proceedings of the Chicago Linguistics Society*, volume 37, pages 133–148.

Dozat, T., Qi, P., and Manning, C. D. (2017). Stanford's graph-based neural dependency parser at the conll 2017 shared task. *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 20–30.

Droganova, K. and Zeman, D. (2017). Elliptic constructions: Spotting patterns in ud treebanks. In *Proceedings of the NoDaLiDa 2017 Workshop on Universal Dependencies (UDW 2017)*, number 135, pages 48–57.

Gulordava, K. and Merlo, P. (2016). Multi-lingual dependency parsing evaluation: a large-scale analysis of word order properties using artificial data. *Transactions of the Association for Computational Linguistics (TACL)*, 4. Presented at ACL 2016.

Johnson, K. (2009). Gapping is not (vp) ellipsis. *Linguistic Inquiry*, 40(2):289–328.

Khoshnavataher, K., Zarrabi, V., Mohtaj, S., and Asghari, H. (2015). Developing monolingual persian corpus for extrinsic plagiarism detection using artificial obfuscation. In *Notebook for PAN at CLEF 2015*.

Nivre, J., de Marneffe, M.-C., Ginter, F., Goldberg, Y., Hajič, J., Manning, C., McDonald, R., Petrov, S., Pyysalo, S., Silveira, N., Tsarfaty, R., and Zeman, D. (2016). Universal dependencies v1: A multilingual treebank collection. In *Proceedings of the 10th International Conference on Language Resources and Evaluation (LREC 2016)*, pages 1659–1666, Portorož, Slovenia. European Language Resources Association.

Popel, M., Žabokrtský, Z., and Vojtek, M. (2017). Udapi: Universal API for universal dependencies. In *Proceedings of the NoDaLiDa 2017 Workshop on Universal Dependencies (UDW 2017)*, pages 96–101.

Schuster, S., Lamm, M., and Manning, C. D. (2017). Gapping constructions in Universal Dependencies v2. In *Proceedings of the NoDaLiDa 2017 Workshop on Universal Dependencies (UDW 2017)*.

Tyers, F. M., Sheyanova, M., and Washington, J. (2017). Ud annotatrix: An annotation tool for universal dependencies. In *Proceedings of the 16th International Workshop on Treebanks and Linguistic Theories*, pages 10–17.

van der Plas, L., Henderson, J., and Merlo, P. (2009). Domain adaptation with artificial data for semantic parsing of speech. In *Proceedings of NAACL HLT 2009: Short Papers*, pages 125–128, Boulder, Colorado, USA, June.

Zeman, D., Popel, M., Straka, M., Hajič, J., Nivre, J., Ginter, F., Luotolahti, J., Pyysalo, S., Petrov, S., Potthast, M., Tyers, F., Badmaeva, E., Gökırmak, M., Nedoluzhko, A., Cinková, S., Hajič jr., J., Hlaváčová, J., Kettnerová, V., Urešová, Z., Kanerva, J., Ojala, S., Missilä, A., Manning, C., Schuster, S., Reddy, S., Taji, D., Habash, N., Leung, H., de Marneffe, M.-C., Sanguinetti, M., Simi, M., Kanayama, H., de Paiva, V., Droganova, K., Martínez Alonso, H., Çöltekin, Ç., Sulubacak, U., Uszkoreit, H., Macketanz, V., Burchardt, A., Harris, K., Marheinecke, K., Rehm, G., Kayadelen, T., Attia, M., Elkahky, A., Yu, Z., Pitler, E., Lertpradit, S., Mandl, M., Kirchner, J., Fernandez Alcalde, H., Strnadova, J., Banerjee, E., Manurung, R., Stella, A., Shimada, A., Kwak, S., Mendonça, G., Lando, T., Nitisaroj, R., and Li, J. (2017). CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 1–19, Vancouver, Canada. Association for Computational Linguistics.

11. Language Resource References

Ginter, F., Hajič, J., Luotolahti, J., Straka, M., and Zeman, D. (2017). *CoNLL 2017 Shared Task - Automatically Annotated Raw Texts and Word Embeddings*. ISLRN `http://hdl.handle.net/11234/1-1989`.

Nivre, J., Agić, Ž., Ahrenberg, L., et al. (2017). *Universal Dependencies 2.1*. ISLRN `http://hdl.handle.net/11234/1-2515`.

Zeman, D., Potthast, M., Straka, M., et al. (2017). *CoNLL 2017 Shared Task System Outputs*. ISLRN `http://hdl.handle.net/11234/1-2424`.