Punctilog: a New Method of Sentence Structure Representation

Ioachim Drugus, Tudor Bumbu, Victoria Bobicev, Victor Didic, Alina Burduja, Alexandr Petrachi, Victoria Alexei

Abstract

We present the experiments on sentence syntactic structure re-codification from dependency grammar to punctilog, a novel methodology of sentence structure representation. Our goal is to create a corpus annotated using this convention; to this end, we re-use the corpora already created by the Universal Dependency project. Several algorithms had been developed for the transformation. We discuss the obtained structures and frame the necessary steps to improve the results.

Keywords: computational linguistics, sentence structure, dependency grammar, punctilog.

1. Introduction

Over the years, numerous methodologies have been used to represent sentence structures in computational linguistics. All these methodologies aimed to capture the most important component: the meaning of the sentence. Some of them were used largely in various projects; some are less known in the research community.

In this paper, we present the work on punctilog, yet another theory of sentence structure representation. Our goal is to create a corpus annotated using this convention; we plan to use the corpora already created by the Universal Dependency project. We developed several algorithms for the transformation and tested them on a small subset of the Romanian corpus. The results of the algorithms are quite different; we discuss the obtained structures and the possible ways to improve the results.

The rest of the paper is organized in the following sections: Section 2 introduces our motivation; the novel methodology named punctilog is described in Section 3; universal dependencies project is presented in Section 4; the proposed re-annotation algorithms are introduced in Section 5; Section 6 presents and analyses the obtained results; Section 7 concludes with a discussion and future work.

2. Motivation

Since the appearance of Computational Linguistics, multiple methodologies have been proposed for the representation of the sentence structure starting with the famous Chomsky grammars [1], tree adjoining grammar [11], link grammar [12], head-driven grammar [10], dependency grammar [3], and others. All developed methodologies had one common problem: they failed to capture the meaning of the text. This work aims to develop a formalism that will help to capture and represent the meaning of a sentence in a logical way.

3. Punctilog

An annotation symbolism referenced as "Punctuation Markup Language" with the abbreviation "PML", was introduced in [6]. However, there is yet another symbolism "Prague markup language" [8] competing for the abbreviation "PML", and there are also other uses of this abbreviation in domains that are far from linguistics. Therefore, a new term "Punctilog" is introduced for the symbolism described in [6]; this term is alternatively used in the paper. The term "punctuation markup language" is still used for the class of all such markup languages but the "PML" acronym is being avoided. Punctilog is one of the markup languages of this class, the one specified in [6], and there can be currently or emerge later many other languages of this class.

The Punctilog markup language consistently uses several punctuation marks according to the defined strict semantics and provides an extension mechanism that allows adding to the language new annotation elements by specifying their semantics according to the format prescribed by the extension mechanism. Since the time when mathematical discourse started making part of the discourses in natural languages, the punctuation marks used by natural languages started being treated as a system that is

extending over time and that needs to be managed. The expected uses of Punctilog are:

(a) To serve as a symbolism for marking up various meanings of expressions in a text;

(b) To assist in the management of the punctuation systems of various languages.

A short account of Punctilog can be given by the example below taken from [6] of a sentence and its Punctilog annotation, which uses all Punctilog's punctuation marks:

„Let's go swimming!" called Ion Chistruiatu, but his team, Gicu and Mihai, was up to it already.

([Let's go swimming!] :((called: (Ion ::Chistruiatu))), (((his: <team>): <Gicu, Mihai>) :(((was :<up to>): it) :already))

There are three bracket types used for Punctilog annotation. Square brackets '[', ']' also called "hard brackets" are used to indicate "direct speech". Angle brackets '<', '>' also called "chevrons" are used to indicate an "individual". Round brackets, also called "parentheses", and in this paper also called "soft brackets" are used to indicate a "constituent".

The round brackets, the parentheses, are used in Punctilog annotation in order to visualize the constituent structure of the sentence [7]. A text may be "ambiguous", i.e. it may be treated as having several interpretations and, accordingly, several constituent structures. To disambiguate a text means to arrange the parentheses according to a certain pattern, a "parenthetical pattern". The term "soft brackets" sounds like an appropriate synonym for the Punctilog's parentheses since these can be arranged and rearranged in different manners to obtain many parenthetical patterns and to choose one or several which are considered the most correct.

A pair of parentheses correctly inserted in a text partially disambiguates the text. Obviously, for full disambiguation, the parentheses should be arranged in such a manner that each pair of balanced parentheses comprise a pair of constituents. The expression "(x

y)" where x and y are constituents can be treated as a binary operation. In Punctilog, this operation is called association operation.

4. Universal Dependencies

Universal Dependencies[1] is one of the largest current projects dedicated to syntactic analysis and creation of the treebanks for multiple languages [2]. Its initial aim was the development of the consistent treebank annotation for many languages, intending to facilitate multilingual parser development and cross-lingual learning. The annotation scheme is based on an evolution of Stanford universal dependencies [3] and Google universal part-of-speech tags [4]. During the last years, multiple researchers joined the project and 202 treebanks in this convention for 114 languages have been released by May 2021 [5].

The annotation is coded in conllu format[2]. This format organizes the whole text by a word per line; each word is accompanied by its id, lemma, part of speech code, the id of a headword, and dependency link label. Figure 1 presents the graphical visualization of the dependency annotation created by conllu format viewer[3].

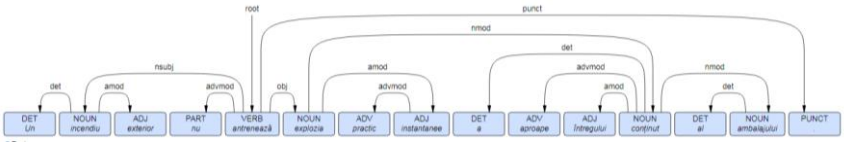

Figure 1. Graphical representation of the dependency annotation for the Romanian sentence "Un incendiu exterior nu antrenează explozia practic instantanee a aproape întregului conținut al ambalajului" (An external fire does not cause a virtually instantaneous explosion of almost the entire contents of the package).

5. Description of the Algorithms

There are several important differences between dependency grammar and punctilog:

[1] https://universaldependencies.org

[2] https://universaldependencies.org/format.html

[3] https://urd2.let.rug.nl/~kleiweg/conllu/

- dependency graph connects words as its nodes with dependency arcs; punctilog forms a tree graph with words as their leaves and intermediate nodes that connect two constituents;
- dependency format allows n-ary connections when several words are connected to one headword; punctilog allows connections only between two elements;
- dependency graph's arcs are labelled by the type of the syntactic relation; punctilog's connections are not named;
- dependency annotation treats punctuation in sentences as tokens the same as words; punctilog ignores all initial punctuation in the sentence.

Due to these differences, the transformation from dependency annotation is not straightforward. Three algorithms have been proposed and developed for the transformation. First of all, the algorithms removed all punctuation from the initial sentence. The next steps consist of successive connections of neighboring words. Each connection forms a constituent that is treated as a new word and can be connected further with another word or constituent.

5.1 Algorithm 1

Each word in the sentence with its headword id is extracted from the conllu format and stored in lists. Then the process of annotation with parenthesis starts. Every two words are taken in the parentheses if the word and its head are the next or the following word. The parentheses are placed repeatedly and the id of the head for the formed constituent is calculated as an average of the heads of the united words. This is done in order to be able to more easily know which constituents are closer to each other.

A while loop is connecting the created constituents to the heads with the closest id number to the dependency averages until only one constituent remains.

5.2 Algorithm 2

After removing punctuation, all words with hyphens before or after are connected with corresponding neighboring words.

Then, connections are performed in two cycles: one from the first word to the last one and another in reverse order: from the last word to the first one. For each word connection is performed if its head is a

neighboring word; the same is applied to the already formed constituents. All formed constituents keep all id of their components and id of their heads. The process of connection stops when only one constituent remains.

5.3 Algorithm 3

This algorithm also connects all words to their neighboring heads to form the constituents. The difference is that the formed constituents obtain the id of the head and its connection; the id of the dependent element is lost. This is why only leaves (words or constituents that are not heads for other elements) are connected.

The connections are performed in two cycles: The inner one checks and connects, if possible, each word from the last word to the first one and the outer one repeats the inner cycle until at least one connection is performed in the inner cycle and stops when no further connections are possible.

6. Comparison of the Algorithms' Results

We had no "gold corpus" annotated absolutely correctly for the algorithm evaluation; we had to evaluate the algorithm manually. 20 sentences annotated in the universal dependencies convention were selected for the evaluation. After its transformation by all three algorithms, we compared the results. Surprisingly, all three algorithms produced quite different results and after closer examination, we concluded that no one of these results was absolutely correct.

Figure 2 presents the same sentence as in Figure 1, but in the form of a dependency tree created by another conllu viewer[4]. The verb "antrenează" is still a root and all other words are connected to it. In order to facilitate the comparison, an online visualization tool for punctilog format has been developed[5]. Figures 3, 4, 5 present graphical representations of the punctilog format created by the algorithms 1, 2, 3 respectively.

In the figures, it is seen that all algorithms produced different structures. All three connected the predicate (*nu antrenează*) to the subject

[4] https://urd2.let.rug.nl/~kleiweg/conllu/

[5] https://univoc.dev/

(*Un incendiu exterior*) and the complex direct object (*explozia practic instantanee a aproape întregului conținut al ambalajului*) was connected partially. Algorithm 1 connected only the first word of the object (*explozia*) to the predicate and the rest of the object was connected apart. Algorithm 2 also connected only a part of the object (*explozia practic instantanee a aproape*) to the predicate and the rest of the subject was taken apart. Algorithm 3 connected the subject and the predicate and the complex object was all together. This variant was the most similar to human judgment. One problem was that in classical structure the object should be connected to the predicate and only after that the subject and the predicate should be united. The other problem of the result was the connections of the complex object elements.

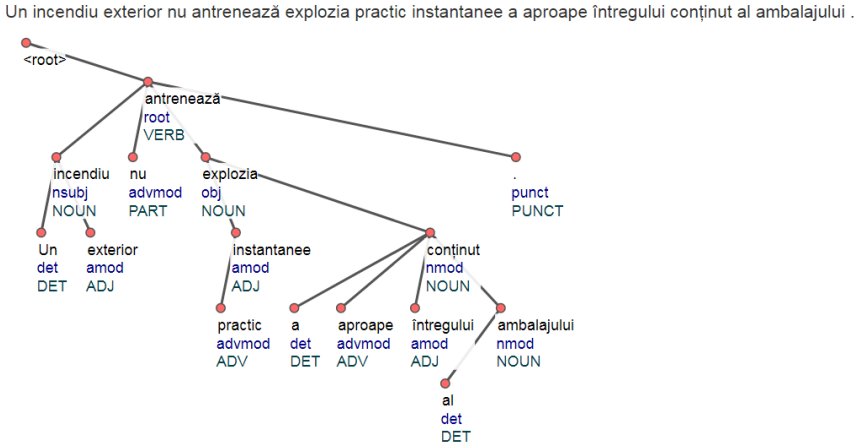

Figure 2. Graphical representation of the universal dependencies annotation of the same sentence as in Figure 1.

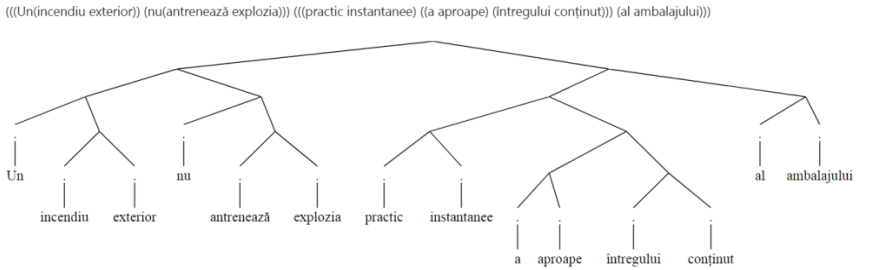

Figure 3. Graphical representation of the punctilog annotation produced by Algorithm 1.

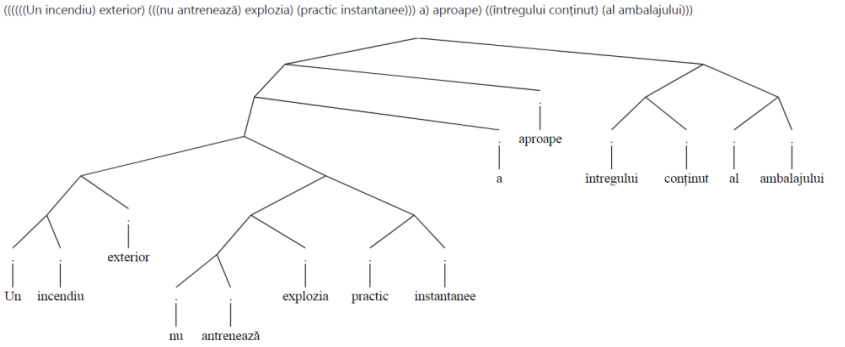

Figure 4. Graphical representation of the punctilog annotation produced by Algorithm 2.

All three algorithms connected this part (*explozia practic instantanee a aproape întregului conținut al ambalajului*) in different ways.

After common discussion between the authors the final and most correct version of the punctilog connections was created; it is presented in Figure 6. The predicate (*nu antrenează*) includes the object (*explozia practic instantanee a aproape întregului conținut al ambalajului*) and all this part is connected to the subject (*Un incendiu exterior*). Complex noun phrase of the object consists of the main part (*explozia practic instantanee*) and the dependent part (*a aproape întregului conținut al ambalajului*) which in turn is formed of parts: (a), (*aproape întregului*) and (*conținut al ambalajului*).

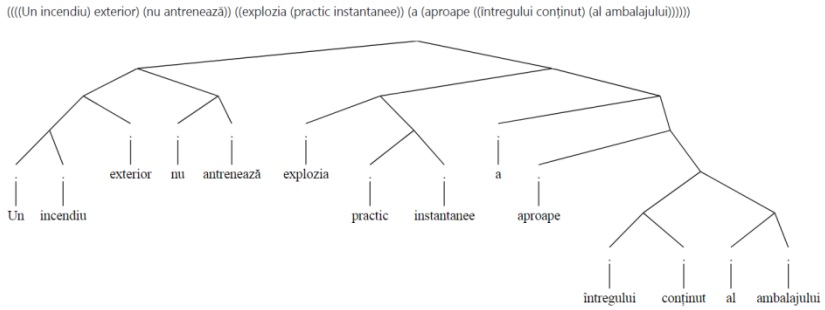

Figure 5. Graphical representation of the punctilog annotation produced by Algorithm 3.

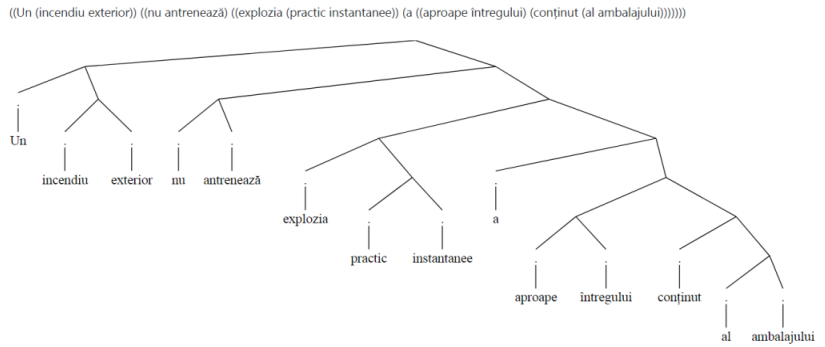

Figure 6. Graphical representation of the punctilog annotation created manually.

7. Discussion and Future Work

As it was discussed in the previous section, we have to modify the algorithms as all of them made different errors in punctilog connections. All algorithms used only the information about the links between words not taking into consideration their morphological and link labels. These labels can be used to connect correctly determiners, adjectives, and articles to the nouns in noun groups; verbs with their auxiliary verbs, adverbs, and particles in verb groups and then to connect formed direct object to predicate and finally subject to the rest of the sentence.

Thus, to form correct punctilog constituents, we have to start with the connections inside classical noun and verb phrases [9]. For noun phrases, firstly the modifiers such as adverbs and adjectives are connected to the words they modify; then articles and parts of complex noun phrases. An example is the complex noun phrase: "*explozia practic instantanee a aproape întregului conținut al ambalajului"* (a virtually instantaneous explosion of almost the entire contents of the package). It is seen in Figures 1 and 2 how the words are connected in dependency grammar convention and their parts of speech. Firstly, adverbs are connected to adjectives, then adjectives to nouns, next articles to the formed groups, and finally several nouns with their dependent words are connected together. The order of connections is:

explozia practic instantanee -> explozia (practic instantanee) -> (explozia (practic instantanee));

a aproape întregului conținut al ambalajului -> a (aproape întregului) conținut al ambalajului -> a (aproape întregului) conținut (al ambalajului) -> a (aproape întregului) (conținut (al ambalajului)) -> (a (aproape întregului) (conținut (al ambalajului))).

Finally, these two parts are connected together. However, there is still possible ambiguity. The part *(a (aproape întregului) (conținut (al ambalajului)))* can be connected as in Figure 6 or as *(a ((aproape întregului) conținut) (al ambalajului))*; this version is presented in Figure 7.

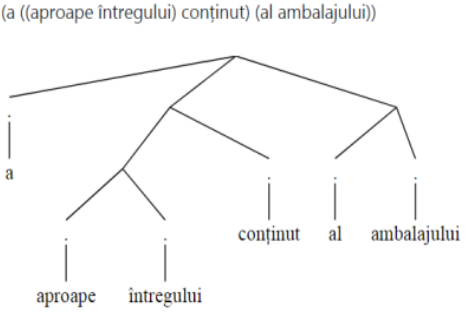

Figure 7. Graphical representation of the alternative punctilog annotation of the complex noun phrase.

The methodology described above in this section is connecting the parts as in Figure 7 as *aproape întregului conținut* is a noun phrase with the main noun and *al ambalajului* is a noun phrase as well. The first word ***a*** with morphological tag *det* and connection to the first noun *conținut* is connected to the first part as well and the structure is slightly different: *((a ((aproape întregului) conținut)) (al ambalajului))*. This structure may also be considered correct.

8. Conclusion

In this paper, ongoing work on the creation of the text corpus annotated with a novel methodology named punctilog is described. The methodology aims to represent the sentence's meaning through its structure. We discuss which structures are appropriate and how to create a corpus of texts annotated in this convention. We present the experiments on the re-annotation of universal dependencies Romanian corpus. We discuss the developed algorithms' results and our future plans for their improvement.

References

[1] N. Chomsky. *The Logical Structure of Linguistic Theory*. Springer, US, 1975, 592 p.

[2] Ryan McDonald, Joakim Nivre, Yvonne Quirmbach-Brundage, Yoav Goldberg, Dipanjan Das, Kuzman Ganchev, Keith Hall, Slav Petrov, Hao Zhang, Oscar Täckström, Claudia Bedini, Núria Bertomeu Castelló, and Jungmee Lee. *Universal Dependency Annotation for Multilingual Parsing*. In: Proceedings of ACL, 2013.

[3] M.-C. de Marneffe, T. Dozat, N. Silveira, K. Haverinen, F. Ginter, J. Nivre, and C.D. Manning. *Universal Stanford Dependencies: A cross-linguistic typology*. In: Proceedings of the 9th International Conference on Language Resources and Evaluation, 2014, pp. 4585-4592.

[4] S. Petrov, D. Das, and R. McDonald. *A Universal Part-of-Speech Tagset.* In: Proceedings of the Eighth International Conference on Language Resources and Evaluation, 2012, pp.2089-2096.

[5] Daniel Zeman, et al. *Universal Dependencies 2.8.1*, LINDAT/CLARIAH-CZ digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University, 2021, http://hdl.handle.net/11234/1-3687.

[6] Ioachim Drugus. *PML: A Punctuation Symbolism for Semantic Markup*. In: Proc. of 11th International Conf. "Linguistic Resources and Tools for Processing the Romanian Language", 2015, pp.79-92.

[7] Andrew Carnie. *Constituent Structure*. Oxford University Press; 2nd edition, 2010, 320 pages, ISBN-10: 0199583463.

[8] Jirka Hana and Jan Štěpánek. *Prague Markup Language Framework*. In: Proceedings of the Sixth Linguistic Annotation Workshop, Association for Computational Linguistics, 2012, pp. 12-21.

[9] J. Miller. *A critical introduction to syntax*. London: Continuum, 2011, 275 pages.

[10] Carl Pollard and Ivan A. Sag. *Head-Driven Phrase Structure Grammar*, Chicago: University of Chicago Press, 1994.

[11] K. Vijay-Shanker and Aravind Joshi. *Unification-Based Tree Adjoining Grammars*, 1991. Technical Reports (CIS).

[12] Daniel Sleator and Davy Temperley. *Parsing English with a Link Grammar*. In: Third International Workshop on Parsing Technologies, 1993.

Ioachim Drugus[1], Tudor Bumbu[1, 3], Victoria Bobicev[2], Victor Didic[2, 3], Alina Burduja[2], Alexandr Petrachi[2], Victoria Alexei[2]

[1]Vladimir Andrunachievici Institute of Mathematics and Computer Science
E-mail: ioachim.drugus@math.md, tudor.bumbu@math.md

[2]Technical University of Moldova
E-mail: victoria.bobicev@ia.utm.md, alina.burduja@iis.utm.md, alexandr.petrachi@iis.utm.md, victoria.lazu@ia.utm.md, victor.didic@iis.utm.md

[3]Est Computer
E-mail: bumbutudor10@gmail.com, victor.didic864@gmail.com