STARK: A Tool for Dependency Tree Extraction and Analysis

Kaja Dobrovoljc[1,2], **Luka Krsnik**[3], **Marko Robnik-Šikonja**[3]
[1]University of Ljubljana, Faculty of Arts
[2]Jozef Stefan Institute, Ljubljana, Slovenia
kaja.dobrovoljc@ff.uni-lj.si
[3]University of Ljubljana, Faculty of Computer and Information Science
krsnik.luka92@gmail.com marko.robnik@fri.uni-lj.si

Relevant UniDive working groups: WG1, WG4

1 Introduction

In addition to the well-established benefits to language technology, syntactically annotated corpora, i.e. treebanks, represent a valuable methodological tool for research in linguistics and other language-based disciplines. This is particularly the case with the cross-linguistically harmonized Universal Dependencies treebank collection (de Marneffe et al., 2021), which introduces many methodological opportunities for research on linguistic diversity and universality.

In line with the growing number of UD treebanks, which currently encompass over 240 treebanks in more than 130 languages, there has also been an increase in the development of tools facilitating their linguistic investigation[1]. Most of these, however, require the users to formulate specific queries based on pre-defined assumptions on the nature and the distribution of structures under investigation, such as specifying the number of nodes occurring in a tree and the relationships among them.

To complement such top-down, deductive, data-informed treebank investigations with bottom-up, inductive, data-driven analysis, we present a recently developed tool for the extraction of dependency trees from Universal Dependencies treebanks.

2 Design and Settings

STARK[2] is an open-source python-based command-line tool which, for a given input treebank in the CONLL-U format, produces a frequency list of dependency trees matching the various user-defined criteria. In addition to the **general settings**, such as the location of the input and output files, the minimum frequency threshold, the maximum number of lines and the statistics to be calculated in the output file, these include:

- **tree size**, which defines the number of nodes in the trees to be extracted (integer or range);
- **tree type**, defining whether all possible subtrees of a given size should be extracted or full subtrees only (values *all* or *complete*);
- **dependency type**, defining if dependency labels should be considered or not (values *labeled* or *unlabeled*);
- **node type**, which defines what level of token information should be considered (values *form*, *lemma*, *upos*, *xpos*, *feats* or *deprel*); and
- **node order**, defining whether trees should be differentiated based on the surface word order or not (values *fixed* or *free*).

Optionally, the users can also introduce additional tree-specific **restrictions**, such as defining a specific set of dependency labels allowed to occur in the tree, or specific constraints on the root node, if, for example, one is interested in extracting trees with nouns as heads only. Last, similarly to the common approach to treebank querying mentioned above, the tool also allows the users to formulate a query defining a specific type of tree to be extracted.

After execution, the tool generates all possible queries based on the user-defined criteria described above, and iterates over all possible trees in the input file to search for input trees that match any of the generated query trees. The matches are returned and printed in an output file described below.

[1]See, for example, the various tools for UD treebank annotation, browsing or visualisation listed on the UD project website: https://universaldependencies.org/.
[2]https://gitea.cjvt.si/lkrsnik/STARK

3 Output

The results are given in the form of a tab-separated file with a list of extracted trees (one per line) sorted by frequency, as illustrated in Table 1. In addition to the structure of the trees[3] and their absolute and relative frequencies in the input treebank, the frequency list also includes other relevant information in relation to specific settings, such as the information on the number of nodes in the tree and their surface order.

Tree structure	Freq.
DET <det NOUN	1345
ADP <case DET <det NOUN	1163
ADP <case NOUN	1090
PRON <nmod:poss NOUN	487
CCONJ <cc NOUN	476

Table 1: An example output illustrating top-most frequent types of noun-headed trees in the English GUM Treebank (`tree_size` 2-10, `tree_type` complete, `dependency_type` labeled, `node_type` upos, `node_order` fixed, `root` upos=NOUN).

As an optional parameter, the statistical association between the nodes of the tree, i.e. the collocational strength (Evert et al., 2008), can also be calculated using several common association measures (MI, MI^3, Dice, logDice, t-score, simple-LL). As illustrated by the example given in Table 2, this is a particularly useful feature for treebank-driven lexical analysis.

Tree structure	MI
Image > (: < Nick > Moreau) > .	37.0
On < the < other < hand > ,	27.3
In < other < words > ,	20.6
As < a < result > ,	19.0
at < the < same < time	18.3

Table 2: An example output showing top-most salient noun-headed trees in the English GUM Treebank (`tree_size` 2-10, `tree_type` complete, `dependency_type` unlabeled, `node_type` form, `node_order` fixed, `root` upos=NOUN, `frequency_threshold` 5; sorted by MI score).

[3]The structure of a tree is described using the expressive dep_search query language (Luotolahti et al., 2015). If word order is selected as a tree-differentiating feature, each output tree is formalised in two ways: using the default, order-agnostic, dep_search-compatible 'free' structure, and using its slightly modified alternative, in which the nodes are listed according to their actual order on the surface, as in Tables 1 to 3 below.

4 Treebank Comparison

By using the optional `--compare` parameter, the tool allows users to compare the obtained results to a separate treebank through the so-called keyness analysis, a common corpus linguistic approach in which the frequency of a given phenomena in one corpus is compared to the frequency of the same phenomena in another corpus by means of a selected statistical measure (Gabrielatos, 2018). In particular, the tool returns the popular LL, BIC, log ratio, odds ratio and %DIFF keyness scores. Table 3 illustrates the output.

Tree structure	LL
PRON <nmod:poss NOUN >case PART	22.8
ADP <case PRON <nmod:poss NOUN <compound NOUN	16.0
SYM <cc NOUN	10.6
ADP <case PRON <nmod:poss NOUN	7.4
DET <det NOUN	4.7

Table 3: An example output showing top-most key types of noun-headed trees in the English GUMReddit Treebank in comparison to the English GUM Treebank (`tree_size` 2-10, `tree_type` complete, `dependency_type` labeled, `node_type` upos, `node_order` fixed, `root` upos=NOUN, `frequency_threshold` 5; sorted by LL).

5 Visualization

Although STARK does not support any visualization of the output trees, the string describing the structure of a tree is directly transferable to the online treebank browsing services adopting the same query language, such as the SETS (Luotolahti et al., 2017)[4] and Drevesnik[5] treebank browsing services. This allowing the users to access specific sentences with extracted tree types in the treebank(s) under investigation. In the future, this aspect of the tool could be further developed by providing conversions to query languages featured in other similar treebank browsing services.

6 Conclusion

We outlined the recently developed STARK open-source tool for dependency trees extraction from universally parsed corpora. Through its wide selection of highly-customizable settings, it facilitates

[4]http://depsearch-depsearch.rahtiapp.fi/ds_demo/

[5]https://orodja.cjvt.si/drevesnik/en/

data-driven linguistic research on various levels of grammatical description (from syntactic to lexical analysis), with varying degrees of granularity (from general patterns to specific structures) and scope (from single treebank analysis to treebank comparisons).

Although the methodological potential of such an approach to treebank exploration is yet to be fully exploited and evaluated (but see Goldberg and Orwant (2013) for a popular general usage application), several potential applications could be envisaged in support of the goals of the UniDive COST Action. For example, the tool could be used to identify low-frequency trees suggesting treebank-specific idiosyncracies (or even annotation inconsistencies, as observed in Table 3); to identify multi-word expressions of various types and lengths; or to identify instances of potential treebank- or language-specific grammatical patterns, to name just a few of the use cases at hand.

Acknowledgements

This work was supported by the project A Treebank Approach to the Study of Spoken Slovenian (Z6-4617) and the research program Language Resources and Technologies for Slovene (P6-0411) funded by the Slovene Research Agency, as well as through the 2019 CLARIN.SI Resource and Service Development grant.

Key References

Marie-Catherine de Marneffe, Christopher D. Manning, Joakim Nivre, and Daniel Zeman. 2021. Universal Dependencies. *Computational Linguistics*, 47(2):255–308.

Stefan Evert et al. 2008. Corpora and collocations. *Corpus linguistics. An international handbook*, 2:1212–1248.

Costas Gabrielatos. 2018. Keyness analysis: Nature, metrics and techniques. In *Corpus approaches to discourse*, pages 225–258. Routledge.

Yoav Goldberg and Jon Orwant. 2013. A dataset of syntactic-ngrams over time from a very large corpus of English books. In *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 1: Proceedings of the Main Conference and the Shared Task: Semantic Textual Similarity*, pages 241–247.

Juhani Luotolahti, Jenna Kanerva, and Filip Ginter. 2017. dep_search: Efficient search tool for large dependency parsebanks. In *Proceedings of the 21st Nordic Conference on Computational Linguistics*, pages 255–258.

Juhani Luotolahti, Jenna Kanerva, Sampo Pyysalo, and Filip Ginter. 2015. SETS: Scalable and efficient tree search in dependency graphs. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations*, pages 51–55, Denver, Colorado. Association for Computational Linguistics.