STARK:
A Toolkit for Dependency (Sub)Tree Extraction and Analysis

Luka Krsnik
Centre for Language Resources and Technologies
University of Ljubljana
Ljubljana, Slovenia
krsnik.luka92@gmail.com

Kaja Dobrovoljc
University of Ljubljana
Jozef Stefan Institute
Ljubljana, Slovenia
kaja.dobrovoljc@ff.uni-lj.si

Abstract

We present STARK, a lightweight and flexible Python toolkit for extracting and analyzing syntactic (sub)trees from dependency-parsed corpora. By systematically slicing each sentence into interpretable syntactic units based on configurable parameters, STARK enables bottom-up, data-driven exploration of syntactic patterns at multiple levels of abstraction—from fully lexicalized constructions to general structural templates. It supports any CoNLL-U-formatted corpus and is available as a command-line tool, Python library, and interactive online demo, ensuring seamless integration into both exploratory and large-scale corpus workflows. We illustrate its functionality through case studies in noun phrase analysis, multiword expression identification, and syntactic variation across corpora, demonstrating its utility for a wide range of corpus-driven syntactic investigations.

1 Introduction

Syntactically annotated corpora, or treebanks, have become indispensable in linguistic research, supporting work on grammar description (Ferrer-i Cancho et al., 2022), typological comparison (Levshina, 2022), genre analysis (Wang and Liu, 2017), as well as language technology development and understanding (Zeman et al., 2018; Lin et al., 2012; Hewitt and Manning, 2019). As their availability grows, so too does the ecosystem of tools designed to facilitate their exploration, most notably through treebank browsing services such as Grew-match (Guillaume, 2021), PML Tree Query (Štepánek and Pajas, 2010), and INESS (Rosén et al., 2012).

Despite this growing infrastructure, most existing tools are inherently *query-based*. They require the user to formulate a specific hypothesis or structural pattern of interest—typically by specifying the number of words involved, their morphological properties, and their syntactic relationships. Such top-down approaches are well-suited for targeted investigation, but they offer limited support for inductive, bottom-up discovery of patterns—particularly in cases where no prior expectations about syntactic configurations are available or desirable.

In practical terms, if a researcher is interested in noun phrase structures, most existing tools will allow them to search for examples of a specific pattern — for example, a noun preceded by an adjectival modifier. However, such tools do not typically support asking what kinds of noun phrase structure patterns actually occur in the corpus, how frequent they are, or whether any rare or unexpected patterns emerge — particularly in contrast to another dataset.

To address this gap, we present STARK (Subtree Analysis and Retrieval Kit), a toolkit for bottom-up, treebank-driven syntactic analysis. Rather than relying on predefined queries, STARK automatically extracts all trees and subtrees that meet general structural criteria specified by the user—effectively slicing a parsed corpus into interpretable syntactic patterns, which can then be counted and compared within or across corpora.

The remainder of the paper introduces STARK's core functionality and configurable parameters (§2), illustrates its analytic capabilities through frequency, association, and comparison outputs (§3), and then details its features for example retrieval and visualization (§4), scalability and performance optimization (§5), and accessibility via an interactive online demo and open-source release (§6).

2 Core Functionality

STARK is an open-source Python toolkit for extracting and analyzing syntactic (sub)trees from dependency-parsed corpora. It operates by systematically slicing each sentence into smaller, interpretable syntactic units based on configurable structural parameters, described below.

2.1 Basic Design

STARK operates on input files in CoNLL-U format, the standard tab-separated format for representing word-level syntactic and morphological annotations in dependency-parsed corpora. Although the tool was developed with the Universal Dependencies (UD) annotation scheme (de Marneffe et al., 2021) in mind, it is not limited to UD-compliant data: it accepts any corpus in CoNLL-U format, regardless of scheme-specific tagsets or label inventories, and handles multi-root sentence structures and other non-canonical phenomena.

To illustrate STARK's core functionality, consider the sentence in Figure 1:

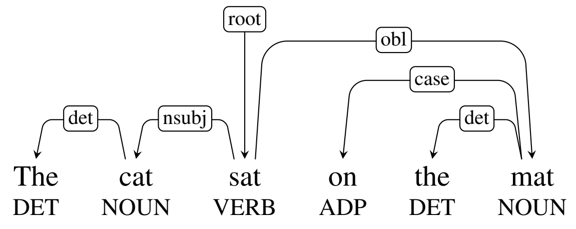


Figure 1: Dependency tree for the sentence *The cat sat on the mat* using the UD annotation scheme.

STARK treats every word in a sentence as a potential syntactic head and extracts the subtree rooted at that word—that is, the head and all its dependents. This yields a collection of overlapping (sub)trees,[1] each capturing a local syntactic configuration. Table 1 shows the resulting structures extracted using this procedure, if we were to extract unlabeled trees with surface word forms only (but see Section 2.2 for more options).

Head	**Subtree**
`The`	`The`
`cat`	`The < cat`
`sat`	`(The < cat) < sat > (on < the < mat)`
`on`	`on`
`the`	`the`
`mat`	`on < the < mat`

Table 1: (Sub)trees extracted from the parsed sentence in Figure 1.

Each unique tree is then counted and written to a tab-separated output file, one per row. Trees are represented using a simplified query-like syntax inspired by `dep_search` tool (Luotolahti et al., 2017),[2] with additional columns optionally showing frequency, statistical scores, or illustrative examples.

We now turn to the main parameters that control how trees are represented, filtered, and extracted.

2.2 Tree Representation

STARK offers several options for determining how trees are constructed and represented in the output.

- **Node type** (`--node_type`) determines what information is used to represent each token. Users can choose any CoNLL-U field, such as surface form, lemma, or UPOS tag, or omit node content entirely. For instance, the tree `the < mat` could appear as `DET < NOUN` (UPOS), or `the < mat` (lemma), depending on the setting.

- **Dependency labels** (`--labeled`) are typically included (e.g., `the <det mat`), but can also be omitted to support more abstract structure. Subtype retention is optional (`--label_subtypes`), allowing users to distinguish between coarse and fine-grained labels (e.g., `nmod:poss` vs. `nmod`).

- **Node order** (`--fixed`) determines whether linear word order is taken into account. When enabled, token order contributes to the identity of the tree; otherwise, trees are treated as equivalent regardless of surface order, which is particularly useful for analyzing languages with flexible constituent structure.

These settings allow users to extract trees at different levels of specificity, from fully lexicalized constructions (e.g., Table 3) to more abstract syntactic patterns (e.g., Table 2 and 4).

2.3 Tree Filtering

Users can further restrict extracted trees using a range of filters:

- **Tree size** (`--size`) specifies the number of nodes in each tree, either as a single value (e.g., 3) or a range (e.g., 2-5). This setting is optional—using a broad range like 1-1000 effectively extracts all trees, regardless of size.

[1] In what follows, we use the term *tree* to refer to both full trees and subtrees, unless otherwise specified.

[2] `A > B` means A governs B; `A < B` means A is governed by B; dependency labels follow the operator (e.g., `A >obj B`); and parentheses (e.g., `A > (B > C)`) mark attachment priority. Letters like A and B stand for tokens, which can be constrained by form, lemma, UPOS, etc. The underscore (_) represents any token.

- **Head constraints** (`--head`) limit tree extraction to structures rooted in tokens matching a specified property, such as `upos=NOUN` or `lemma=want`. This is useful for focusing on specific construction types (e.g., noun-headed phrases, as in Table 2), or for studying lexicogrammatical behavior of individual words.

- **Label constraints** combine two parameters (`--allowed_labels` and `--ignored_labels`) to restrict which relations can appear in a tree. Users can specify a whitelist of allowed relations (e.g., `nsubj|obj|iobj`), or indicate relations to ignore as irrelevant (e.g., `punct`), without discarding the tree itself.

- **Custom queries** (`--query`) provide fine-grained control by allowing users to specify an exact dependency pattern to match.[3] Crucially, STARK applies all other representation settings (see Section 2.2) when generating the output, enabling hybrid workflows that combine top-down targeting with bottom-up extraction—e.g., listing all lexical realizations or surface order permutations of a pattern (as in Table 4, for example).

These flexible and combinable filters give users precise control over the granularity and scope of extraction, making STARK adaptable to a wide range of research goals.

2.4 Optional Processing Mode

By default, STARK extracts full subtrees rooted at each token—i.e., the head and all direct/indirect dependents—producing syntactically coherent units. The `--complete` parameter can be adjusted to instead extract all connected subtrees anchored at a token, including partial or nested fragments. While this mode can reveal finer combinatorial detail, it is computationally more demanding and best suited for small datasets or targeted analyses.

3 Statistical Analysis

In addition to extracting and representing syntactic structures, STARK provides a range of statistical measures that support quantitative corpus-based syntactic analysis. These include basic frequency counts, association scores, and keyness comparisons, all computed based on the extracted trees. In this section, we illustrate each type of analysis on different corpora and configurations to highlight STARK's flexibility.[4]

3.1 Frequency

By default, STARK outputs absolute and relative frequency counts for each extracted tree. Relative frequencies are normalized per million tokens, enabling comparison across corpora of different sizes. This information is useful for identifying both dominant constructions and rare syntactic patterns (including annotation mistakes), providing insight into the overall distribution of specific structures in a corpus. For example, Table 2 in Appendix A lists the ten most frequent noun-headed trees in the English GUM UD Treebank (Zeldes, 2017), revealing the most common types of nominal phrase patterns in the language that can inform descriptive grammar work and usage-based models, or serve as a basis for comparative studies.

3.2 Association

In addition to frequency, STARK optionally computes several statistical association measures via the `--association_measures` parameter. These quantify the strength of co-occurrence between nodes within a tree (Evert, 2009) and include mutual information (MI), MI^3, Dice, logDice, t-score, and log-likelihood (LL). The scores are particularly useful for identifying collocationally strong structures, especially in lexicalized output. This is illustrated in Table 3 in Appendix A, which lists the top ten noun phrases of size 3 or more in the French GSD UD treebank (Guillaume et al., 2019) ranked by logDice, revealing a range of nominal multi-word expressions.

3.3 Keyness

STARK supports keyness analysis via the `--compare` parameter, which compares extracted trees against a reference corpus. It calculates the relative frequency of each tree in both corpora and computes several keyness scores (Gabrielatos, 2018), including log-likelihood (LL), BIC, log ratio, odds ratio, and %DIFF. These help detect struc-

[3]Currently, queries are written in dep_search (e.g. '`upos=VERB >nsubj _ >obj _`' for verb-subject-object trees retrieved in Table 4), but support for other query languages like Grew (Guillaume, 2021) or Semgrex (Bauer et al., 2023) could be added in future.

[4]Due to space constraints, more information on the specific measures is available in the cited literature and at: `https://github.com/clarinsi/STARK/blob/master/statistics.md`.

tures that are disproportionately frequent or underrepresented in one corpus relative to another, making this feature particularly useful for comparing syntactic or lexical behavior across genres, domains, or languages.

Table 4 in Appendix A illustrates this by comparing subject–verb–object (SVO) patterns in the spoken (SST) and written (SSJ) Slovenian UD treebanks (Dobrovoljc et al., 2017; Dobrovoljc and Nivre, 2016), highlighting constructions that are more or less prominent in speech in comparison to writing.

4 Example Retrieval and Visualisation

STARK offers optional output enhancements to support qualitative analysis and visualization. Users can retrieve a sample sentence per tree (`--example`), with marked nodes, or add node-level (`--node_info`) and head-level (`--head_info`) details for further analysis.

STARK also supports integration with online treebank browsing services. If the input is an official UD treebank (i.e., follows the standard naming convention), enabling the `--grew_match` option generates clickable links to the corresponding patterns in the Grew-match interface (Guillaume, 2021).[5] These links let users explore all instances of a given tree in context within the latest UD release and leverage additional Grew-match functionalities.

For compatibility with legacy tools, the `--depsearch` option outputs trees in the `dep_search` syntax used by earlier platforms such as SETS (Luotolahti et al., 2015), dep_search (Luotolahti et al., 2017) or Drevesnik (Štravs and Dobrovoljc, 2024).[6]

5 Scalability

STARK has been tested on all official UD treebanks and can handle corpora of various sizes and annotation styles, including multi-root sentences and non-standard labels. Output volume can be managed via frequency thresholds (`--frequency_threshold`) or by capping the number of output trees (`--max_lines`), making it easy to scale STARK to large datasets while maintaining interpretability.

Several advanced settings have also been introduced to further improve performance and scalability. These include multi-core support (`--cpu_cores`), internal caching for repeated experiments (`--internal_saves`), and chunked processing of directory-based corpora (`--continuation_processing`). Users can also select between two extraction modes: the default `greedy_counter`, optimized for bottom-up tree extraction, and the `query_counter`, which performs better when used with specific target patterns.

6 Availability and Online Demo

STARK is freely available as an open-source tool under the Apache 2.0 license.[7] In addition to the command-line interface, the tool is also released as a Python library via PyPI (`pip install stark-trees`),[8] enabling seamless integration into custom scripts and larger NLP workflows. Comprehensive documentation is available through the GitHub and PyPI repositories where users can find detailed explanations of all parameters, usage examples, and configuration tips.

To further support accessibility, STARK is also available via an interactive online demo.[9] The web interface covers all core functionalities of the tool, allowing users to select a treebank, configure extraction settings (see Section 2), and explore the output in an interactive table view. Unlike the command-line version, the online interface also provides visualisations for one or multiple example instances of the tree.

As such, the online demo is particularly useful for exploratory browsing, classroom use, and first-time users unfamiliar with the command-line interface. Screenshots of both the settings panel and the output view are shown in Figures 2 and 3 in Appendix B.

7 Conclusion

We introduced STARK, a versatile toolkit for bottom-up syntactic analysis of dependency-parsed corpora. By extracting, ranking, and comparing syntactic (sub)trees, it enables exploratory and data-driven research without requiring predefined queries. The tool supports a wide range of configurations and outputs, and is available as a command-line tool, Python library, and online demo.

Its practical value has already been demonstrated through early adoption in a range of re-

[5]https://universal.grew.fr

[6]https://orodja.cjvt.si/drevesnik/

[7]https://github.com/clarinsi/STARK

[8]https://pypi.org/project/stark-trees/

[9]https://orodja.cjvt.si/stark/

search contexts, from integration into tools such as the DELTA diversity pipeline (Estève and Dobrovoljc, 2025) and ComparaTree treebank comparison tool (Terčon and Dobrovoljc, 2025), to studies on syntactic profiling of spoken data (Hüll and Dobrovoljc, 2025; Dobrovoljc, 2025), learner essays (Munda and Holdt, 2025), and parallel multilingual corpora (Čibej, 2025).

Acknowledgments

This work was financially supported by the Slovenian Research and Innovation Agency through the research projects Treebank-Driven Approach to the Study of Spoken Slovenian (Z6-4617), Large Language Models for Digital Humanities (GC-0002), and the research program Language Resources and Technologies for Slovene (P6-0411). Special thanks go to Marko Robnik Šikonja for his early contributions to the project, and to CLARIN.SI for additional support through the 2019 and 2024 Resource and Service Development grants. We also thank Tina Munda, Nives Hüll, Luka Terčon, and Louis Estève for their help with testing and feedback. Generative AI tools were employed to support language editing. The authors retain full responsibility for the content, analysis, and interpretation presented in this work.

References

John Bauer, Chloé Kiddon, Eric Yeh, Alex Shan, and Christopher D. Manning. 2023. Semgrex and ssurgeon, searching and manipulating dependency graphs. In *Proceedings of the 21st International Workshop on Treebanks and Linguistic Theories (TLT, GURT/SyntaxFest 2023)*, pages 67–73, Washington, D.C. Association for Computational Linguistics.

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

Kaja Dobrovoljc. 2025. Counting trees: A treebank-driven exploration of syntactic variation in speech and writing across languages. *Preprint*, arXiv:2505.22774.

Kaja Dobrovoljc, Tomaž Erjavec, and Simon Krek. 2017. The Universal Dependencies treebank for Slovenian. In *Proceedings of the 6th Workshop on Balto-Slavic Natural Language Processing*, pages 33–38, Valencia, Spain. Association for Computational Linguistics.

Kaja Dobrovoljc and Joakim Nivre. 2016. The Universal Dependencies treebank of spoken Slovenian. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 1566–1573, Portorož, Slovenia. European Language Resources Association (ELRA).

Louis Estève and Kaja Dobrovoljc. 2025. A new pipeline for measuring diversity across various linguistic levels. Abstract accepted at the UNIDIVE 3rd General Meeting in Budapest.

Stefan Evert. 2009. *58. Corpora and collocations*, pages 1212–1248. De Gruyter Mouton, Berlin, New York.

Ramon Ferrer-i Cancho, Carlos Gómez-Rodríguez, Juan Luis Esteban, and Lluís Alemany-Puig. 2022. Optimality of syntactic dependency distances. *Phys. Rev. E*, 105:014308.

Costas Gabrielatos. 2018. *Keyness Analysis: nature, metrics and techniques*, pages 225–258. Routledge, United Kingdom.

Bruno Guillaume. 2021. Graph matching and graph rewriting: GREW tools for corpus exploration, maintenance and conversion. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: System Demonstrations*, pages 168–175, Online. Association for Computational Linguistics.

Bruno Guillaume, Marie-Catherine de Marneffe, and Guy Perrier. 2019. Conversion et améliorations de corpus du français annotés en Universal Dependencies. *Revue TAL : traitement automatique des langues*, 60(2):71–95.

John Hewitt and Christopher D. Manning. 2019. A structural probe for finding syntax in word representations. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4129–4138, Minneapolis, Minnesota. Association for Computational Linguistics.

Nives Hüll and Kaja Dobrovoljc. 2025. Word order variation in spoken and written corpora: A cross-linguistic study of svo and alternative orders. In *Proceedings of the SyntaxFest 2025*. To appear.

Natalia Levshina. 2022. Corpus-based typology: applications, challenges and some solutions. *Linguistic Typology*, 26(1):129–160.

Yuri Lin, Jean-Baptiste Michel, Erez Lieberman Aiden, Jon Orwant, Will Brockman, and Slav Petrov. 2012. Syntactic annotations for the google books ngram corpus. In *Proceedings of the ACL 2012 System Demonstrations*, ACL '12, page 169–174, USA. Association for Computational Linguistics.

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, Gothenburg, Sweden. Association for Computational Linguistics.

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.

Tina Munda and Špela Arhar Holdt. 2025. First insights into the syntax of slovene student writing: A statistical analysis of šolar 3.0 vs. učbeniki 1.0. In *Proceedings of the SyntaxFest 2025*. In print.

Victoria Rosén, Koenraad De Smedt, Paul Meurer, and Helge Dyvik. 2012. An open infrastructure for advanced treebanking. In *META-RESEARCH Workshop on Advanced Treebanking at LREC2012*, pages 22–29. Hajič, Jan.

Jan Štepánek and Petr Pajas. 2010. Querying diverse treebanks in a uniform way. In *Proceedings of the 7th International Conference on Language Resources and Evaluation (LREC 2010)*, pages 1828–1835.

Miha Štravs and Kaja Dobrovoljc. 2024. Service for querying dependency treebanks drevesnik 1.1. Slovenian language resource repository CLARIN.SI.

Luka Terčon and Kaja Dobrovoljc. 2025. Comparatree: A multi-level comparative treebank analysis tool. In *Proceedings of the SyntaxFest 2025*. To appear.

Yaqin Wang and Haitao Liu. 2017. The effects of genre on dependency distance and dependency direction. *Language Sciences*, 59:135–147.

Amir Zeldes. 2017. The GUM corpus: Creating multilayer resources in the classroom. *Language Resources and Evaluation*, 51(3):581–612.

Daniel Zeman, Jan Hajič, Martin Popel, Martin Potthast, Milan Straka, Filip Ginter, Joakim Nivre, and Slav Petrov. 2018. CoNLL 2018 shared task: Multilingual parsing from raw text to Universal Dependencies. In *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 1–21, Brussels, Belgium. Association for Computational Linguistics.

Jaka Čibej. 2025. A computational method for analyzing syntactic profiles: The case of the elexis-wsd parallel sense-annotated corpus. In *Proceedings of the SyntaxFest 2025*. To appear.

A Example STARK Outputs

Rank	Tree	Freq.	Example
1	NOUN	4436	*They're bringing* ***drugs****.*
2	DET <det NOUN	2331	*Plant* ***the cuttings****.*
3	ADP <case DET <det NOUN	1874	*Remove* ***from the oven****.*
4	ADP <case NOUN	1815	*Prepare* ***for impact****.*
5	CCONJ <cc NOUN	809	*Distinguish concepts* ***and prototypes****.*
6	PRON <nmod:poss NOUN	803	***My house*** *was empty and cold.*
7	ADJ <amod NOUN	735	*We have* ***hard work*** *ahead.*
8	ADP <case ADJ <amod NOUN	602	*They show it* ***in many ways****.*
9	DET <det ADJ <amod NOUN	576	*Yeah, or turn* ***a deaf ear****.*
10	ADP <case PRON <nmod:poss NOUN	569	*He was recognized for some* ***of his books****.*

Table 2: Top 10 nominal phrase structures in the English GUM UD Treebank sorted by frequency. STARK settings used: `node_type = upos, labeled = yes, label_subtypes = yes, fixed = yes, size = 1-10000, head = upos=NOUN, complete = yes`.

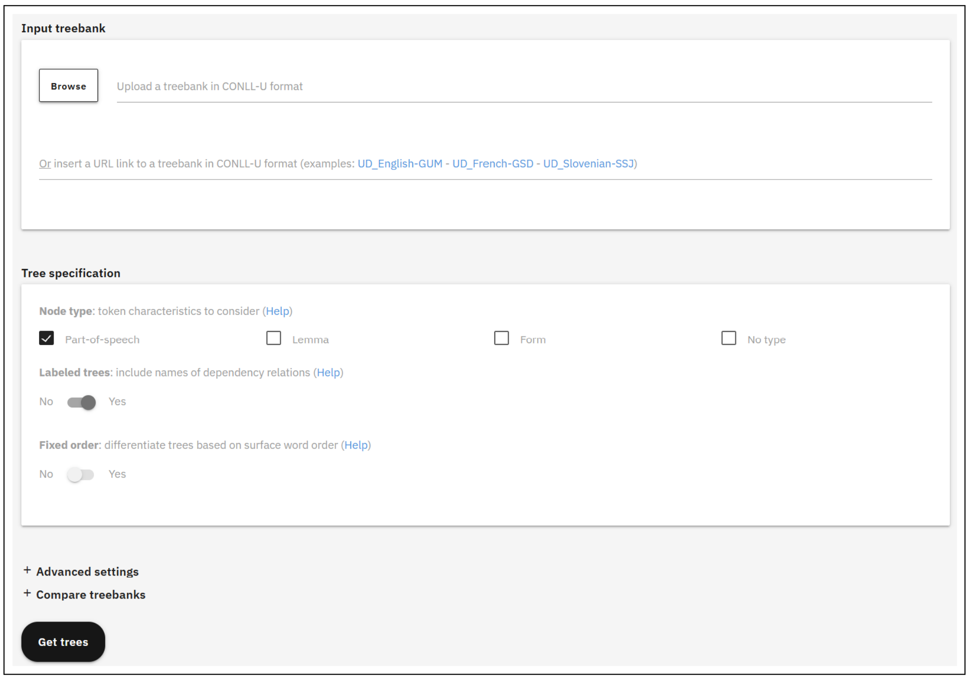
Rank	Tree	Freq.	No. of nodes	logDice
1	qualité > / < prix	5	3	9.35
2	pour < la < première < fois	35	4	7.50
3	une < nouvelle < fois	10	3	6.97
4	de < le < monde	92	3	6.78
5	pour < sa < part	7	3	6.70
6	par < la < suite	24	3	6.58
7	sur < des < prises	5	3	6.19
8	d' < araignées > aranéomorphes	7	3	6.12
9	l' < année > suivante	9	3	6.02
10	de < la < ville	48	3	6.00

Table 3: Top 10 nominal multi-word expressions in the French GSD UD Treebank ranked by logDice association measure. STARK settings used: `node_type = form, labeled = no, fixed = yes, size = 3-10, head = upos=NOUN, complete = yes, association_measures = yes, frequency_threshold = 5`.

Rank	Tree	RF in SST	RF in SSJ	OR	Example
1	_ <nsubj _ <obj _	1768.4	1639.9	1.08	*če naši* ***možje to naredijo****.*
2	_ <obj _ <nsubj _	1321.2	1299.2	1.02	***tega nihče*** *ni* ***razumel*** *dolgo.*
3	_ >nsubj _ >obj _	396.4	408.1	0.97	*pa* ***imam jaz*** *tudi svoje* ***obveznosti****.*
4	_ >obj _ >nsubj _	203.3	325.7	0.62	***zanimala*** *sta* ***vas novinarstvo*** *in filozofija*
5	_ <obj _ >nsubj _	1473.7	3159.9	0.47	***kaj izraža*** *ta* ***glagol*** *?*
6	_ <nsubj _ >obj _	3323.4	8225.5	0.40	*katera* ***črta razpolavlja kot*** *?*

Table 4: SVO patterns in the spoken Slovenian SST UD Treebank ranked by Odds Ratio (OR) keyness measure, when compared to the written SSJ UD Treebank. RF = relative frequency. STARK settings used: `labeled = yes, fixed = yes, query = upos=VERB >nsubj _ >obj _, complete = no, compare = sl_ssj-ud.conllu`.

B STARK Web Interface



Input treebank

Browse — Upload a treebank in CONLL-U format

Or insert a URL link to a treebank in CONLL-U format (examples: UD_English-GUM - UD_French-GSD - UD_Slovenian-SSJ)

Tree specification

Node type: token characteristics to consider (Help)

☑ Part-of-speech ☐ Lemma ☐ Form ☐ No type

Labeled trees: include names of dependency relations (Help)

No [✓ Yes]

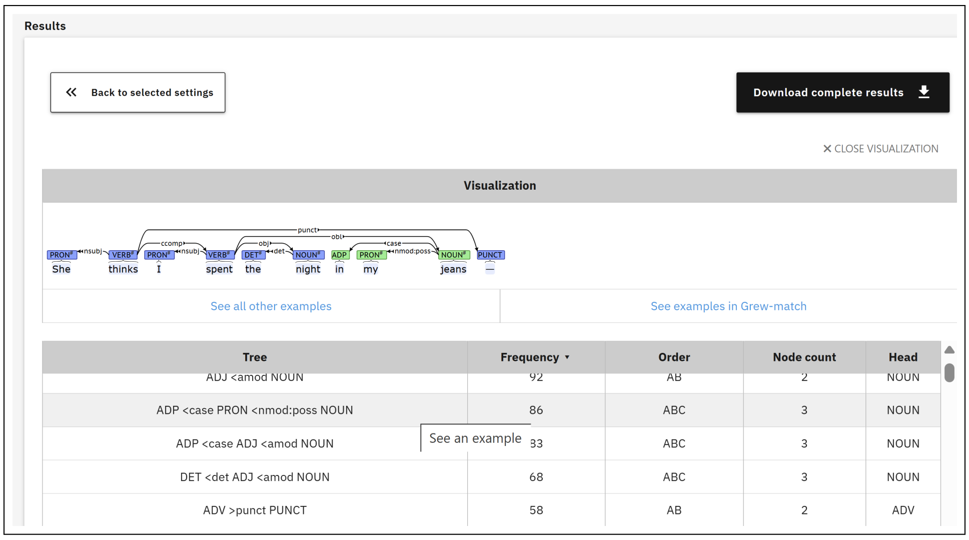
Fixed order: differentiate trees based on surface word order (Help)

[✓ No] Yes

+ Advanced settings
+ Compare treebanks

Get trees

Figure 2: Screenshot of the STARK online demo interface, showing the interactive settings selection, from basic tree specification to advanced tree filtering and treebank comparison options.



Results

Back to selected settings

Download complete results

✕ CLOSE VISUALIZATION

Visualization

See all other examples — See examples in Grew-match

Tree	Frequency ▾	Order	Node count	Head
ADJ <amod NOUN	92	AB	2	NOUN
ADP <case PRON <nmod:poss NOUN	86	ABC	3	NOUN
ADP <case ADJ <amod NOUN	See an example 33	ABC	3	NOUN
DET <det ADJ <amod NOUN	68	ABC	3	NOUN
ADV >punct PUNCT	58	AB	2	ADV

Figure 3: Screenshot of the STARK online demo interface, showing the interactive results table, an example tree visualisation, and links to explore all matched examples in both the demo and Grew-match.