UDeasy: a Tool for Querying Treebanks in CoNLL-U Format

Luca Brigada Villa
University of Bergamo / Pavia
luca.brigadavilla@unibg.it

Abstract

Many tools are available to query a dependency treebank, but they require the users to know a query language. This paper presents UDeasy, an application whose main goal is to allow the users to easily query and extract patterns from a dependency treebank in CoNLL-U format. To do this, users are prompted in a series of dialogs to enter relevant information about syntactic nodes, their properties, relationship, and positions.

Keywords: dependency treebanks, query tool

1. Introduction

CoNLL-U is the standard format for the annotation of dependency treebanks in many frameworks such as Universal Dependencies (UD) (de Marneffe et al., 2021) and Surface Syntactic Universal Dependencies (SUD) (Gerdes et al., 2018). It is a revised version of the CoNLL-X format (Buchholz and Marsi, 2006) and consists of ten fields separated by single tab characters carrying information about the morphology and the syntax of each token.

In this paper, I present UDeasy, a tool whose goal is to make it easy to design a query for treebanks annotated in CoNLL-U format. The paper is structured as follows: in Section 2, I list some of the available tools for processing and querying treebanks annotated in CoNLL-U format; in Section 3, I present UDeasy and how to use it; finally, Section 4 contains a summary of the advantages of using UDeasy for quantitative linguistic research.

2. Tools

Among the available tools that allow querying a dependency treebank, it is worth to mention CoNLL-U viewer, UDAPI, TüNDRA and Grew-match. I will discuss them in more detail pointing out the advantages and disadvantages of their use.

2.1. CoNLL-U viewer

CoNLL-U viewer (developed by Milan Straka and Michal Sedlák)[1] is a browser-based visualization tool for CoNLL-U files. It shows the trees representing the sentences stored in a CoNLL-U file uploaded by the users and allows downloading the generated image files.

2.2. UDAPI

UDAPI (Popel et al., 2017) is an API for processing Universal Dependencies. It is available in Python, Perl and Java as a library and, in addition, it can be used from the command-line interface. It allows the users to do operations such as parsing sentences, visualizing trees both in ASCII and HTML, querying treebanks and convert from one format to another.

2.3. TüNDRA

TüNDRA (Martens, 2013) is a web application for querying and visualizing treebanks. It allows to access more than 400 treebanks (most of them dependency treebanks) already available on the website and lets users upload their own treebanks in TCF or CoNLL-U format. Its query language is based on the TIGERSearch language (König and Lezius, 2003). In addition, TüNDRA allows the users to gather statistical information about the results of a query.

2.4. Grew-match

Grew-match (Guillaume, 2021) is a web application for searching graph patterns in treebanks in projects such as Universal Dependencies, Surface Syntax Universal Dependencies, French Sequoia corpus (Candito and Seddah, 2012), three corpora in AMR (Banarescu et al., 2013) and MultiWord Expression annotation from the Parseme project (Ramisch et al., 2020). Grew-match has also an offline version which can be used to query patterns from treebanks owned by the users.

3. UDeasy

UDeasy is an application written in Python 3 with a graphic interface built using the GUI toolkit wxPython[2]. The functions to extract the occurrences from a treebank rely on the udapi Python package (Popel et al., 2017). It has been developed to work on Windows, MacOS and Linux systems. It is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License and published at https://unipv-larl.github.io/udeasy/.

3.1. Why UDeasy

When using one of the applications or tools mentioned earlier, a user may encounter some issues:

[1] https://universaldependencies.org/conllu_viewer.html

[2] see http://wxPython.org/

Proceedings of the CMLC-10 Workshop (@LREC 2022), pages 16–19
Marseille, 20 June 2022

- some of the tools only allow querying treebanks that are hosted online and not uploaded by the users (e.g. Grew-match online)
- some of the tools are designed to be used from a command line interface or included in a script (e.g. UDAPI)
- all the tools force the users to learn a query language

These factors may complicate the work of a linguist who wants to follow a quantitative and data-driven approach.
The goal of UDeasy is to overcome these issues by allowing the users to extract patterns from dependency treebanks with a simplified process. The main advantages of using UDeasy are the following:

- it accepts all the treebanks that are formatted in CoNLL-U
- it has a graphical interface that guides the users step by step in the design of the query
- there is no need to learn any query language

3.2. How to use UDeasy

The tool is designed as a series of panels dedicated to the different parameters that the users may want to set in order to extract a pattern from a treebank. When opening the application, a window pops up and the users are asked to select a CoNLL-U file stored on their computer.

3.2.1. Naming the nodes

When clicking on the button to confirm the selection of the CoNLL-U file, the nodes panel appears. The users are asked to give a name to the nodes that are involved in the target pattern. The names are not part of the actual query, but they will be used to refer to the target nodes in the subsequent steps.

3.2.2. Selecting the features for each target node

In the panel that appears, the users can indicate one or more features that the target nodes involved in the pattern must match. The users can select any of the CoNLL-U fields (`lemma`, `upos`, `deprel`) or any of the sub-features that some CoNLL-U fields have such as `feats` and `misc`.
As values for the selected features, the users can either enter one or more values that have to be matched. If the users enter a value, then the feature must have that exact value for the node if the parameter `value is` is selected; if more values are passed, they have to be written between squared brackets and separated by commas. If the feature has one of those values, then the node is included in the results.
In the feature dropdown menu, the users will find all the CoNLL-U fields and some of the sub-features of `feats` and `misc`: if they want to look for a sub-feature not included in the menu, they can insert the value with the keyboard.

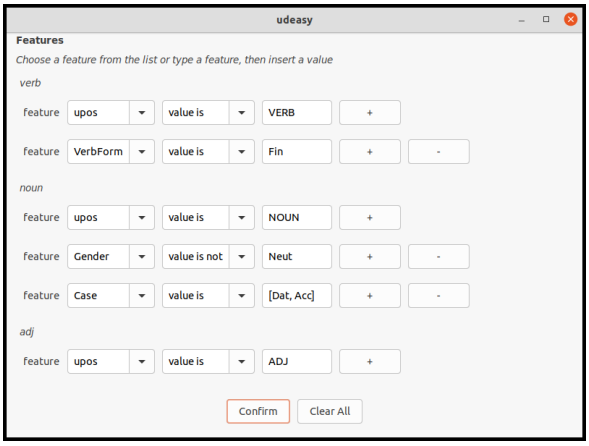

Figure 1: The features panel how it appears in the application; *verb*, *noun* and *adj* are the names given to the nodes in the previous stage (see Section 3.2.1).

3.2.3. Specifying the relations among nodes

In order to specify the relations among the nodes, the users can select from the dropdown menu in the relations panel the nodes (using the names given to the target nodes in the first panel - see Section 3.2.1) and a relation selected from `is parent of`, `is ancestor of` and `is sibling of`.

3.2.4. Specifying the relative positions among nodes

The last parameter the users might want to specify is the relative positions of the nodes. Like the relations, the conditions for the relative positions must involve two nodes. If the users do not want to specify any ordering among the nodes, they can leave the fields empty. Otherwise, they must give a value for the first three fields, i.e. the nodes and the ordering relation (`precedes` or `follows`). In addition to that, they can specify a distance between the nodes selecting either `by exactly` or `by at least` in the fourth field and entering an integer number.

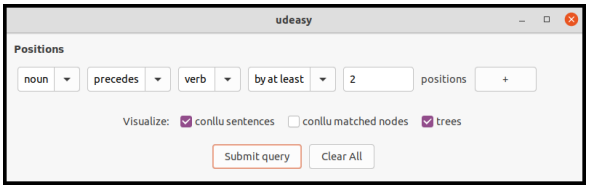

Figure 2: The positions panel as it appears in the application.

3.2.5. Results

As shown in Figure 2, the users can select some visualization options such as `conllu sentences`, `conllu matched nodes` and `trees` according to whether they want to see in the results the sentences that have at least a matched pattern formatted in

CoNLL-U, the nodes involved in the matched patterns and the trees of such sentences.
According to what the users have selected, the results will appear in a new window after clicking the button `Submit query`.

3.2.6. Statistics

In the window where the results appear, the users will see an option named `Stats` which allows getting some statistical information about the patterns matched by the submitted query such as word order, distances among the nodes and the distribution of the values of features.

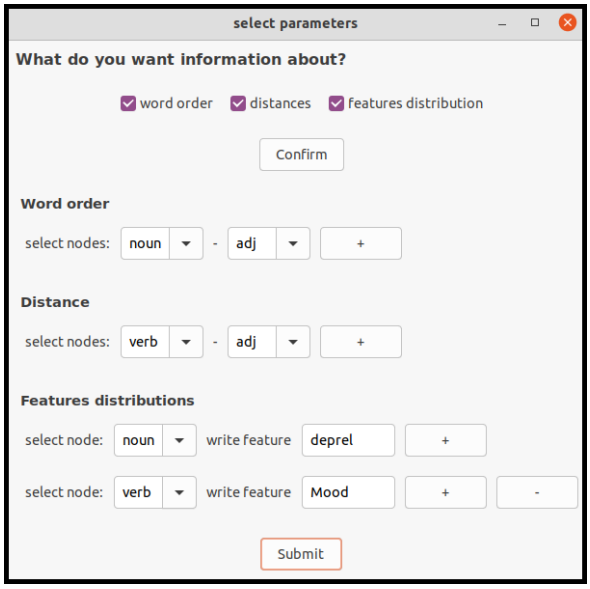

Figure 3: The statistics panel as it appears in the application.

For example, considering the parameters shown in Figure 3, the users will obtain the ordering of the nodes they named *noun* and *adj*, the distribution of the distances between the nodes *verb* and *adj* along with their average distance in the matched sentences and the distribution of the selected features of the nodes *noun* and *verb*.

Mood	count	frequency
Ind	219	0.706452
Sub	85	0.274194
Imp	6	0.0193548

Figure 4: The table showing the values of the feature *Mood*.

For the case of the feature *Mood*, the output will be a table showing all the possible values this feature can take with their frequency in the results, as shown in Figure 4.

4. Conclusion and Future Work

As shown in Section 3, UDeasy is a user-friendly tool that can be used without any knowledge of programming or query languages. I believe it would be a useful tool for the community of linguists who want to use a data-driven approach and for the students who approach dependency treebanks without almost any experience with queries.
Future work might include the implementation of additional features such as new visualization options for the results or new statistics obtainable by the users. Additionally, UDeasy may be upgraded allowing the users to process treebanks formatted in CoNLL-U Plus[3] and to include regular expressions in their queries.

5. Bibliographical References

Banarescu, L., Bonial, C., Cai, S., Georgescu, M., Griffitt, K., Hermjakob, U., Knight, K., Koehn, P., Palmer, M., and Schneider, N. (2013). Abstract Meaning Representation for sembanking. In *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*, pages 178–186, Sofia, Bulgaria, August. Association for Computational Linguistics.

Buchholz, S. and Marsi, E. (2006). CoNLL-X shared task on multilingual dependency parsing. In *Proceedings of the Tenth Conference on Computational Natural Language Learning (CoNLL-X)*, pages 149–164, New York City, June. Association for Computational Linguistics.

Candito, M. and Seddah, D. (2012). Le corpus Sequoia : annotation syntaxique et exploitation pour l'adaptation d'analyseur par pont lexical. In *TALN 2012 - 19e conférence sur le Traitement Automatique des Langues Naturelles*, Grenoble, France, June.

de Marneffe, M.-C., Manning, C. D., Nivre, J., and Zeman, D. (2021). Universal Dependencies. *Computational Linguistics*, 47(2):255–308, 07.

Gerdes, K., Guillaume, B., Kahane, S., and Perrier, G. (2018). SUD or Surface-Syntactic Universal Dependencies: An annotation scheme near-isomorphic to UD. In *Universal Dependencies Workshop 2018*, Brussels, Belgium, November.

Guillaume, B. (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, April. Association for Computational Linguistics.

König, E. and Lezius, W., (2003). *TIGERSearch User's Manual IMS*. University of Stuttgart, Stuttgart, Germany.

Martens, S. (2013). Tündra: A Web Application for Treebank Search and Visualization. In *Proceedings*

[3] https://universaldependencies.org/ext-format.html

of The Twelfth Workshop on Treebanks and Linguistic Theories (TLT12), pages 133–144, Sofia, Bulgaria, December.

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, Gothenburg, Sweden, May. Association for Computational Linguistics.

Ramisch, C., Savary, A., Guillaume, B., Waszczuk, J., Candito, M., Vaidya, A., Barbu Mititelu, V., Bhatia, A., Iñurrieta, U., Giouli, V., Güngör, T., Jiang, M., Lichte, T., Liebeskind, C., Monti, J., Ramisch, R., Stymne, S., Walsh, A., and Xu, H. (2020). Edition 1.2 of the PARSEME shared task on semi-supervised identification of verbal multiword expressions. In *Proceedings of the Joint Workshop on Multiword Expressions and Electronic Lexicons*, pages 107–118, online, December. Association for Computational Linguistics.