Turk J Elec Eng & Comp Sci
() : –

A collaborative grammar annotation tool for agglutinative languages

Salih Furkan Akkurt[1,*], **Büşra Marşan**[2], **Susan Uskudarli**[1]
[1]Department of Computer Engineering, The Faculty of Engineering, Boğaziçi University, Istanbul, Turkey
[2]Department of Linguistics, The Faculty of Arts and Sciences, Boğaziçi University, Istanbul, Turkey

Received: .202 • **Accepted/Published Online:** .202 • **Final Version:** ..202

Abstract:

The significance of high quality treebanks has been steadily increasing due to their crucial role in developing natural language processing tools. The creation of such treebanks is an enormously labor-intensive and time-consuming task due to the need for qualified annotators and the quantity of data to be annotated. Tools that support this process have been developed, however, most of them are not suitable for annotating large treebanks of agglutinative languages such as Turkish. Agglutinative languages require additional features for their annotation compared to the ones without agglutination. They also require extensive annotation of their morphology and have longer dependency arc lengths, making the annotator focus essential to the process. Low-resource agglutinative languages need suitable tools that will help their annotators to create high quality resources. This paper describes the design and implementation of a web-based collaborative grammatical annotation tool for annotating large treebanks, which is also suitable for agglutinative languages. This work builds on extensive experience with a previously developed grammatical annotation tool to annotate a Turkish treebank. The main objectives of the tool are to: (1) support the creation of valid and consistent annotations, (2) increase the speed of the annotation process, (3) improve the user experience of annotators, (4) support collaboration among annotators, and (5) provide an open-source and easily deployable web-based annotation tool with a flexible application programming interface (API) to benefit the scientific community. This paper discusses the requirements elicitation, design, implementation, and evaluation of BOAT-v2 along with examples.

Key words: natural language processing, linguistic annotation, annotation tool, web application, dependency parsing, Universal Dependencies

1. Introduction

Treebanks are important resources in the development of natural language processing (NLP) tools. Quality NLP tools need treebanks that are manually annotated by linguistic experts. This is especially true for agglutinative languages due to their complex morphologies. The creation of such treebanks is highly labor-intensive and time-consuming due to the meticulous attention required. Thus, annotation tools that support this process are essential.

In recent years, there have been significant efforts to bridge the gap in data resources available for agglutinative low-resource languages. Grammatical annotation involves annotating each token of a sentence with corresponding linguistic values. Universal Dependencies (UD) [26] is the most widely accepted standard for annotations of grammar. The annotation format of UD called CoNLL-U (Computational Natural Language Learning) defines a set of linguistic tags for grammatical annotation. For example, universal part of speech

*Correspondence: furkan.akkurt@boun.edu.tr

(UPOS) tag is used for annotation of a token's part of speech. The tag for annotation of morphological features (FEATS) represents additional lexical and grammatical properties of tokens which are not covered by other tags. Specifically for agglutinative languages, the FEATS tag is very frequently annotated with multiple values due to their complex morphology. This makes the effort required to annotate agglutinative languages significantly higher. Annotation tools with drag-drop and mouse-based interfaces, although appealing, are not well suited for agglutinative languages as they require alternating among input modalities frequently, disrupting the flow.

BOAT-v1 [24] is an annotation tool that was developed to support annotation of morphologically rich languages (MRLs) to produce treebanks compliant with the UD framework [26]. The experience during the use of it revealed several points of improvement for such annotation tools. The main takeaway was a much better understanding of the time, effort, cognitive load, and extra information requirements of the annotation process. Improvements regarding these aspects should, consequently, produce higher quality data resources.

This work presents a web-based collaborative grammatical annotation tool (BOAT-v2) that has been designed based on the experience with BOAT-v1. [1] In light of the feedback from this experience, we wanted the tool to be a web application that supports multiple users to enable a collaborative environment for annotators. Several user experience improvements were implemented to enhance the flow of an annotation session. The design and implementation of the tool aimed to: (1) support creating valid and consistent annotations with increased speed, (2) significantly improve the user experience of annotators, (3) allow collaboration among annotators during the annotation process, and (4) provide an open-source and easily deployable web-based annotation tool with an API to benefit the scientific community. The development started with requirements elicitation, for which earlier experiences and in-depth interviews with an experienced annotator were taken into account.

The resultant tool has been evaluated with positive feedback. This feedback indicates that the ability to collaborate within the tool is needed to increase the efficiency of multi-annotator treebank creation. The final version is available on Boğaziçi University's NLP platform [4] and provided as an open-source resource.

The main contributions of this work are:

- Design of a grammatical annotation tool based on requirements elicited from experienced annotators who are linguists,
- The development of a tool that takes into account annotators' experience to improve resulting annotations,
- Multi-user support to provide individual spaces for annotations, computation of inter-annotator agreements, and other potential collaboration,
- The development of a web-based annotation tool based on a supporting API for programmatic access and extensibility, and
- Packaging of the tool to support easy access by virtualizing it using Docker [7] and providing the code as an open-source resource.

The remainder of this paper is organized as follows: Section 2 lays the background of the proposed tool, Section 3 presents related work, Section 4 describes the requirements and design, Section 5 presents the implementation and new features, Section 6 presents a use case of annotation, and Section 8 provides a discussion along with our conclusions.

[1] An earlier version of this paper was presented at the The International Conference on Agglutinative Language Technologies as a challenge of Natural Language Processing (ALTNLP) [1] in 2022.

2. Background

2.1. BoAT-v1

BoAT-v1 [24] is a standalone tool for annotating treebanks that are compatible with the UD framework [26], implemented using Python [15] and Qt [20]. It was specifically developed for annotating Turkish treebanks and is particularly suitable for agglutinative languages; however, it can be used for other languages as well. It supports annotation of a single treebank at a time. Annotations are stored in a file in CoNLL-U format. The file is updated during the annotation process. It uses a validation script developed by UD to dynamically display errors to the annotator as they annotate. BoAT-v1 was used to create the BOUN Treebank [23] – a manually annotated Turkish dependency treebank comprising 9,761 sentences from 5 different domains: essays, national newspapers, instructional texts, popular culture articles and biographical texts.

For annotation of any sentence, an annotator is shown a table which has a token per row with its corresponding tags (ID, FORM, LEMMA, UPOS, XPOS, FEATS, HEAD, DEPREL, DEPS, and MISC as detailed in [24]). The annotator manually enters values for each tag of each token. It supports splitting and joining of lemmas which is particularly significant for agglutinative languages, since tokens in agglutinative languages are comprised of many affixes whose annotations require distinct lemmas. Tokens that are split result in additional two rows whose lemmas are then manually adjusted by an annotator such that they become distinct parts of a token. Figure 2 in Appendix A shows annotation of a sentence. The token "yoktu" (ID: 4-5) is split into "yok" (ID: 4) and "tu" (ID: 5). Furthermore, it parses the FEATS tag's value into individual morphological features. The number of morphological features are much higher in agglutinative languages, therefore the FEATS tag often consists of numerous distinct features. As such, they are difficult to read. BoAT-v1 supports viewing these features individually under their corresponding columns (e.g. Case=Nom—Number=Sing—Person=3 can be shown in columns "Case", "Number", and "Person" with the values "Nom", "Sing", and "3"). It also allows the annotators to be able to take notes for specific annotations.

Comparison of UD Treebanks			
Treebank	**Non-unique feats**	**Total tokens**	**Feats per token**
UD_Turkish-ATIS	112214	45875	2.446
UD_English-ATIS	63434	61879	1.025
UD_Turkish-PUD	32583	16536	1.970
UD_English-PUD	23995	21176	1.133

Table 1. Comparison of morphological feature annotation in UD treebanks

3. Related Work

Annotation tools may be characterized in terms of their accessibility and the support they provide for different languages, various annotation categories, user interface modalities, annotation related standards, multiple annotators, searching of annotations, and automation. Adherence to standards is recommended to get the most benefit from the annotated data sources. Universal Dependencies [26] is an actively growing standard for annotation of treebanks that intends to cover all human languages and its support for agglutinative languages is evolving.

Many dependency annotation tools have been proposed with many of them being UD-compliant, such as brat [17], UD Annotatrix [25] and DGA [2]. Almost all tools are capable of annotating a variety of languages and all the tools that are detailed below do so. They mostly require mouse-based user interactions, which is

inefficient for annotation of agglutinative languages due to the need for extensive annotation for most tokens.

ITU Treebank Annotation Tool [14] is an open-source standalone tool with 3 versions. It was developed specifically for Turkish. It has three stages of annotation: morphological analysis, morphological disambiguation and syntax analysis. It offers semi-automated support for annotators through analyzers for creating new datasets as well as correcting already existing Turkish treebanks. It mostly relies on mouse-based interactions and doesn't support the UD framework. This tool has been used to annotate the ITU Web Treebank [22]. Its latest version (3) was written in Java, requiring Java to be installed on the system used for annotations.

brat [17] is a web-based *text* annotation tool which provides a collaborative annotation environment. Being web-based, it's usable across platforms. It's very flexible in its annotation types and has support for many annotation tasks such as dependency, entity or coreference annotation. It's also used for visualization of annotations. Annotators use mouse and keyboard together to annotate. Annotators are able to import and export their annotations in a file format called *standoff* specific to brat. It is configurable to support annotations compliant with UD and is used for visualization in the UD documentation system. brat supports real-time collaboration, even allowing annotators to edit the same annotation. It supports searching the documents or annotations of them with various filters.

The Dependency Grammar Annotator [2] (also known as DGA or DgAnnotator) is a dependency annotation tool. It enables its users to annotate POS and dependency relation tags via mouse clicks. It supports transfer of annotations in UD compliant formats. It doesn't support multiple annotators to work on the same treebank. There is no search functionality. It was written in Java [12], and consequently requires Java.

WebAnno [27] is a web-based open-source annotation tool, that is not restricted to dependency annotations but has support for morphological, syntactical, and semantic annotations also, with multi-user support. To annotate morphological features of a token, it requires several mouse clicks which is impractical for MRLs. The sentences to be annotated are displayed vertically in a single screen, unlike many tools that focus on a single sentence at a time. It supports curation of annotations created by multiple annotators. It has two installment options, standalone and server. The standalone version requires Java, while the server version requires a Tomcat [18] server with a MySQL [13] database.

UD-Annotatrix [25] is a UD-specific dependency annotation tool that aims to be simpler than brat. Mouse usage is necessary, in addition to keyboard entries, for annotation. It also supports real-time collaboration if the tool is deployed in a server. It employs annotation standards of UD and CG3 [21] It doesn't have a search functionality.

CoNLL-U Editor [11] is a web-based annotation tool, specific to the CoNLL-U [5] format of UD. It saves every edit of annotations by version control. If specific lists of part-of-speech tags are provided by the annotators, it supports autocompletion during annotation. Multiple annotators are able to annotate the same treebank, if they don't work on the same sentence at a time. Annotators can search using any combination of linguistic tags of UD using the tool. It requires a Java-based server.

StarDust [28] is a standalone dependency annotation tool. It uses 2 layers: morphological disambiguation and dependency annotation layers, sequentially. In the dependency annotation layer, annotators are expected to click on a dependeny word and drag the cursor to the head word. It stores each sentence's annotation in a different file. For backups of the annotations, it uses Dropbox [8], which is a proprietary cloud storage platform. There were many treebanks annotated with this tool: 1 English and 5 Turkish treebanks.

BoAT-v2 is the second iteration of BoAT-v1. It improves on it by including a search functionality, a database to represent sentences in a more essential way rather than a plaintext file and an accessible web

interface with an API for flexibility. BOAT-v2 has been developed to reduce clutter that BOAT-v1 was reported to have in parts of the screen in the feedback of the previous BOAT-v1 usage.

Comparison of Annotation Tools					
Tool	**Mouse**	**Keyboard**	**Standards**	**Multi-user**	**Search**
brat	+	+	Extendable	+	+
UD Annotatrix	+	+	UD & CG3	+	-
DgAnnotator	+	-	UD	-	-
CoNLL-U Editor	+	+	UD	+	+
ITU	+	+	UD	-	-
WebAnno	+	+	UD	+	-
StarDust	+	+	-	-	-
BOAT-v2	+	+	UD	+	+

Table 2. Properties of above-mentioned annotation tools

4. Requirements and Design

Requirements elicitation and validation meetings were held with annotators who worked on the BOUN Treebank involving thousands of sentences to identify the software requirements. The main requirements that emerged are:

Collaborative annotation: The sheer sizes of treebanks necessitate that they are annotated by multiple annotators. The tool must support multiple annotators working on the same treebank. The ability to share annotation experiences is vital for reference and consistency. Hence, the tool must provide means for registering multiple annotators and keeping track of their respective annotations.

Search: During the annotation of a sentence, annotators sometimes need to refer to previous annotations for guidance. An annotator should be able to search a treebank according to surface forms, and importantly, according to linguistic features. They should be able to perform complex searches that involve several criteria such as UD tags, individual morphological features of the FEATS tag, and surface forms. This requirement reduces cognitive load as well as facilitates consistency among annotations.

Keyboard-oriented input: Annotations of agglutinative languages require annotation of several morphological features for many tokens of a sentence in comparison to analytical languages like English. While drag-and-drop interfaces can work well for analytical languages, they don't for MRLs as they require more toggles between mouse and keyboard interactions which becomes frustrating. Keyboard-based interaction must be offered for all possible tasks.

Support for sentence annotation: To support speedy and accurate annotations, *autocompletion* should be supported whenever possible. Due to typologies of and syncretism being generally higher in MRLs, automatic parsing of such treebanks is difficult and they more often than not fail to create valid parsings of tokens. Thus, refining automatic parsing becomes an important task for agglutinative languages. Words in MRLs tend to have morphemes stacked on roots. Annotators must be able to refine or correct automatically parsed entries, which for agglutinative languages includes splitting or joining of lemmas.

Use of screen real estate and customization: Annotators annotate many sentences each session. For agglutinative languages, sentences tend to be long and complicated. The view of such sentences can cover a considerable part of the screen. The annotator must focus on the sentence being annotated, the lemmas, and all the features, which requires much concentration. The annotation process must not involve

scrolling, if possible. For very long sentences, this may not be possible; however, the sentence being annotated should never go out of vision. Furthermore, every annotator has their unique preferences as to how they do annotation. They must have some control regarding showing or hiding certain information according to their preferences. Overall efficient use of the screen real estate is important to provide the context needed to annotate long sentences.

Given these requirements, we decided that a web-based application that supports multiple authenticated users would support a collaborative platform for annotators. The treebank and user annotations will persist in a database which makes managing the data and searches much more reliable and efficient. Also, an API is developed to support flexibility and extensibility. Finally, in addition to making the tool open-source, we containerized the application using Docker [7] to support its accessibility.

5. Implementation

The annotation tool is implemented using Python [15], the web application development framework *Django* [6] and the API framework *Django REST Framework* (DRF) [9]. The webpages use Bootstrap [3]. PostgreSQL [19] is used as a database management system. The models reflect the UD format of sentences. Most of the sentence annotation functionalities are similar to BOAT-v1. User input is validated according to UD and errors are reported on the annotation page. Three alternative forms of dependency graph visualizations are supported, two of which are newly added, compact and horizontal to reduce the required screen real estate [10, 16].

The following features have been implemented to support the creation of valid annotations with increased speed in a collaborative manner:

Treebank handling: The tool should support the annotation of multiple treebanks. BOAT-v2 uses a database to persist the annotations of multiple annotators of multiple treebanks.

Sentence annotation: An annotator selects the sentence they want to annotate. The sentence annotation page is very similar to BOAT-v1. It consists of three main parts: (1) A table with rows for every token in a sentence and columns, which represent the UD tags of a sentence, corresponding to their annotations; (2) the dependency graph of the sentence; and (3) errors from validation according to the UD framework. The dependency graph and errors are synchronized with the annotations. Several dependency graph presentations are supported to suit the annotator's preference. 2 horizontal dependency graphs are supported since vertical graphs can consume a significant amount of screen real estate, which can lead to loss of focus in long sentences common in agglutinative languages. One vertical dependency graph resembling a tree is also supported in case the annotator prefers it. This vertical graph was also supported in BOAT-v1.

An annotator may need to stop the annotation of a sentence for some reason (i.e. complexity or external interruption). To capture the state of an annotation, status is introduced in BOAT-v2 with values of "New", "Draft" and "Complete". The status of a sentence that has not been annotated is "New". An annotator can set the status to "Draft" or "Complete".

The annotator is able to perform almost all operations, more than what BOAT-v1 allowed, via keyboard action, based on the demand of the annotators (see Section 4 for details). Upon the experience of annotating a Turkish treebank with BOAT-v1, the annotators have reported being very pleased with the convenience and speed resulting from keyboard-based interaction.

Searching for reference and consistency: A search functionality is introduced in BOAT-v2. With it, users are able to search for previously annotated sentences based on combinations of surface text, UD tags, and

individual morphological features. Without a good search feature, an annotator would have to manually search their CoNLL-U files for relevant cases (e.g. how to annotate a certain surface form's UPOS tag). The form might have been inconsistently annotated, alas it is unlikely that a manual search would reveal this problem. In such a case, the annotator would likely use the first encountered usage as a reference. The situation gets more complicated for syncretic morphemes such as *-ki* in Turkish. For example, in the sentence "Evdeki halılar yıkandı." (*The rugs at home were washed.*), the *-ki* acts as an adjectivizer. However, in "Benim halılarım yün, Ayşeninkiler sentetik." (*My rugs are woolen. Ayşe's are synthetic.*), it is pronominal. Searching for sentences where the *-ki* morpheme occurs via basic text search of a file would be hopeless as there would be too many hits since they occur very frequently, owing to its part in the grammar and brevity as a word.

To facilitate effective searching, we have implemented search functionality based on combinations of text and UD tags. Regular expression-based search is also supported. This feature supports annotators to share experiences, which consequently is expected to result in more accurate and consistent treebanks.

Inter-annotator agreement: The consistency of annotations among annotators is an indicator of the quality of the resulting resource. Inter-annotator agreement computes the consistency among annotators. Since this tool keeps track of annotator actions, unlike its predecessor, such computations are straightforward. Some visualizations shall accompany these statistics.

6. Using BoAT-v2

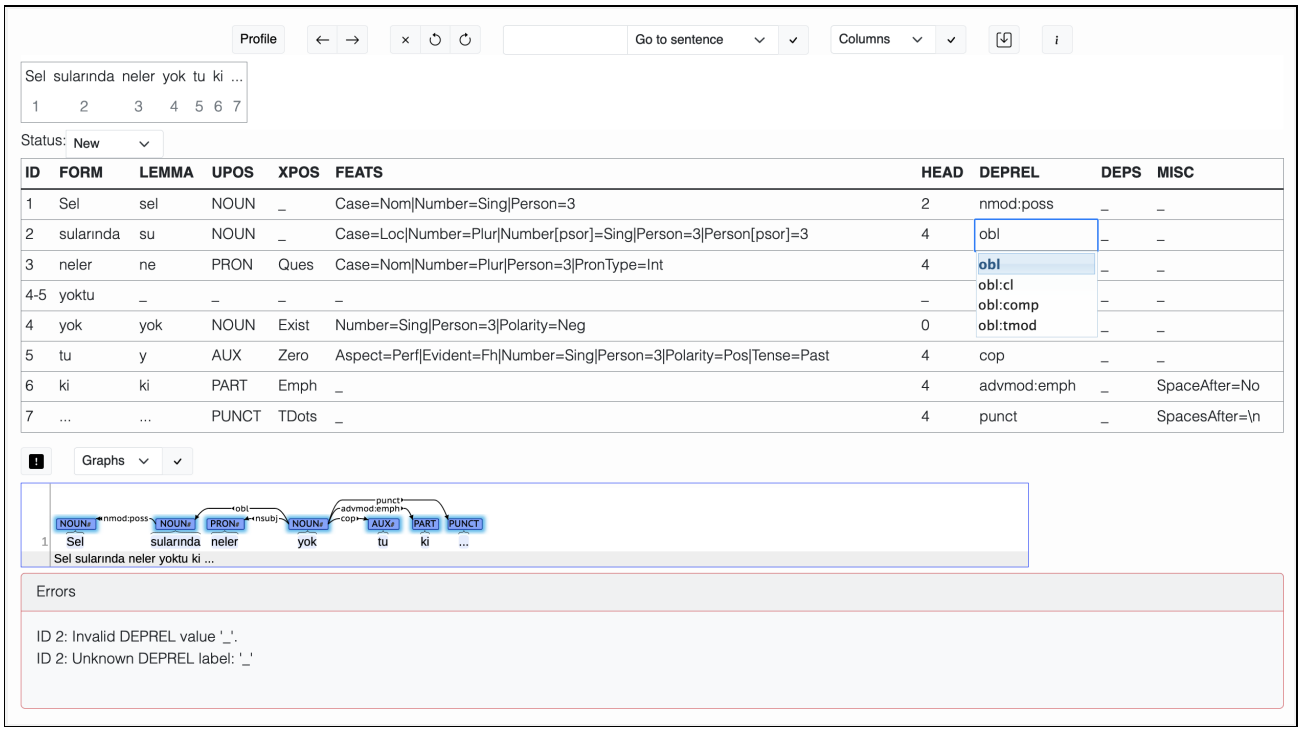

Figure 1. The annotation screen captured while the sentence "Sel sularında neler yoktu ki..." is being annotated. The DEPREL tag for the surface form "sularında" is being annotated by selecting among the valid alternatives that appear in the pop-up. Selections can be made with the use of arrow keys.

A typical annotation can proceed as follows: An annotator selects a sentence from a treebank. An annotation table appears with the sentence parsed according to the UD format. Each row corresponds to a token and its annotations. Figure 1 shows the annotation view while an annotator is annotating a Turkish sentence "Sel sularında neler yoktu ki..." (translation: *What wasn't in the flood waters...*).

An annotator can make use of dependency graphs, errors, and search during annotation. Dependency graphs are visual cues for how lemmas are dependent upon one another. Errors are helpful reminders compliant with UD.

Annotators can customize the columns and dependency graph in accordance with their preferences. When an annotation is finished, its status can be set to "Complete". The annotator can search for previously made annotations in combinations of text and feature values to refer to previously made annotations.

7. Evaluation

We compared the annotation of several sentences with the same number of tokens using BOAT-v1 and BOAT-v2. While we kept the number of words in a sentence constant, we did not use the same sentences since having previously annotated a sentence would impact the annotation on another version. Keeping the number of words the same provides a somewhat comparable experience. There was a noticeable speedup (approximately 30%) using BOAT-v2. Among the new features that are most appreciated are autocompletion, condensed dependency tree representation, significant reduction on scrolling, keyword search, and search by morphological features. The non-search related features are instrumental to retaining focus.

We designed a test case for user testing in July 2022. This test case included testing various functionalities of the tool, such as creating treebanks, annotation of both simple and complex sentences and searching annotations previously created. Simple sentence annotation refers to sentences with less than 10 tokens. Complex sentence annotation included long sentences which also required splitting lemmas. We have performed this test case with 3 linguists and gathered their feedback afterwards. The test results were mostly matching what the expected results were. We have worked on improving the tool further and making it more robust, according to the feedback.

We did testing with several test cases via Selenium in September 2022. In these tests, we aimed to test if the tool's interface works as expected in various use cases, leveraging assertions in the tests. These use cases included logging in, creating and deleting treebanks, and searching sentences. The actual results of these tests were matching the expected results.

We designed another test with several test cases for testing the intuitiveness of the tool and its usefulness in educating new annotators in December 2022. We executed this test with 2 graduates of the Linguistics Department at Boğaziçi University in Istanbul, Turkey. These 2 testers had not had any experience in annotation. We gathered feedback after they have done the test cases we have given them.

8. Discussion and Conclusions

BOAT-v2 aims to extend the functionality of BOAT-v1 as a collaborative web-based application to support the annotation process based on previous experiences. We developed a web application that supports agglutinative languages as described in Section 5.

The implementation choices served our goals well. We believe that having experts in linguistics and experienced annotators in agglutinative treebank creation was instrumental in understanding the requirements and the design process. We held numerous elicitation interviews and further meetings for clarifications and feedback requests.

We used modern software development tools and management practices during the development lifecycle of this tool. The development of an API enables various extensions of this tool and access to the treebanks. The containerization with Docker [7] has facilitated easy delivery and deployment. It is available on Boğaziçi University's NLP platform [4] as an open-source resource. The availability of the tool as a user as well as a developer is valuable for future use and developments.

We are encouraged by the evaluation performed on this tool and anticipate its extensions. In fact, our implementation of BoAT-v2 resulted in a revision request for BoAT-v1 to include the focus enhancing features. This resulted in significant speedup and improved annotator experience, which was reported as a qualitative observation by an experienced annotator.

The feedback coming from the evaluation presents many ways that the tool can be improved. For the tool's future, we plan on maintaining the current functionality, implementing annotation-centric new features, such as automated annotation of certain tags based on rules input by annotators of a treebank, or creating notes on annotations to communicate with other annotators through the application. We also plan to implement support for various languages regarding autocompletion and annotation validation.

Acknowledgment

This work was supported by Boğaziçi University Research Fund Grant Number 16909.

References

[1] Salih Furkan Akkurt, Büşra Marşan, and Susan Uskudarli. BoAT v2 – A Web-Based Dependency Annotation Tool with Focus on Agglutinative Languages, 2022. URL https://arxiv.org/abs/2207.01327.

[2] Giuseppe Attardi. DgAnnotator, 2022. URL http://medialab.di.unipi.it/Project/QA/Parser/DgAnnotator/. [Online; last accessed 27 March 2022].

[3] Bootstrap. Bootstrap v5.1 Documentation, 2022. URL https://getbootstrap.com/docs/5.1. [Online; last accessed 5 May 2022].

[4] Boğaziçi University. TABILAB Repository Home, 2022. URL https://tulap.cmpe.boun.edu.tr. [Online; last accessed 24 April 2022].

[5] Sabine Buchholz and Erwin Marsi. 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 2006. Association for Computational Linguistics. URL https://aclanthology.org/W06-2920.

[6] Django Software Foundation. Django documentation, 2022. URL https://docs.djangoproject.com/en/4.0. [Online; last accessed 25 March 2022].

[7] Docker Inc. Home - Docker, 2022. URL https://www.docker.com. [Online; last accessed 2 May 2022].

[8] Dropbox, Inc. Dropbox.com, 2022. URL http://dropbox.com. [Online; last accessed 18 November 2022].

[9] Encode OSS Ltd. Django REST framework, 2022. URL https://www.django-rest-framework.org. [Online; last accessed 27 March 2022].

[10] Explosion. spaCy, 2022. URL https://spacy.io. [Online; last accessed 27 March 2022].

[11] Johannes Heinecke. ConlluEditor: a fully graphical editor for Universal Dependencies treebank files. In *Universal Dependencies Workshop 2019*, Paris, 2019. URL https://github.com/Orange-OpenSource/conllueditor/.

[12] Oracle. Oracle Java Technologies — Oracle, 2022. URL https://www.oracle.com/java/technologies/. [Online; last accessed 16 December 2022].

[13] Oracle. MySQL, 2022. URL https://www.mysql.com/. [Online; last accessed 16 December 2022].

[14] Tuğba Pamay, Umut Sulubacak, Dilara Torunoğlu-Selamet, and Gülşen Eryiğit. The annotation process of the ITU web treebank. In *Proceedings of The 9th Linguistic Annotation Workshop*, pages 95–101, Denver, Colorado, USA, June 2015. Association for Computational Linguistics. . URL https://aclanthology.org/W15-1610.

[15] Python Software Foundation. Welcome to python.org, 2022. URL https://www.python.org. [Online; last accessed 24 April 2022].

[16] Sampo Pyysalo. conllu.js, 2022. URL http://spyysalo.github.io/conllu.js. [Online; last accessed 25 March 2022].

[17] Pontus Stenetorp, Sampo Pyysalo, Goran Topić, Tomoko Ohta, Sophia Ananiadou, and Jun'ichi Tsujii. brat: a web-based tool for NLP-assisted text annotation. In *Proceedings of the Demonstrations Session at EACL 2012*, Avignon, France, April 2012. Association for Computational Linguistics.

[18] The Apache Software Foundation. Apache Tomcat®, 2022. URL https://tomcat.apache.org/. [Online; last accessed 16 December 2022].

[19] The PostgreSQL Global Development Group. PostgreSQL, 2022. URL https://www.postgresql.org/docs. [Online; last accessed 27 March 2022].

[20] The Qt Company. Qt — Cross-platform software development for embedded & desktop, 2022. URL https://www.qt.io/. [Online; last accessed 25 May 2022].

[21] Tino Didriksen. Constraint Grammar Manual, 2022. URL https://visl.sdu.dk/cg3/single/. [Online; last accessed 18 November 2022].

[22] Dilara Torunoğlu-Selamet, Tuğba Pamay, Umut Sulubacak, and Gülşen Eryiğit. The annotation process of the itu web treebank. 01 2015. .

[23] Utku Türk, Furkan Atmaca, and Şaziye Betül Özateş and Gözde Berk and Seyyit Talha Bedir and Abdullatif Köksal and Balkız Öztürk Başaran and Tunga Güngör and Arzucan Özgür. UD Turkish BOUN, 2020. URL https://universaldependencies.org/treebanks/tr_boun/index.html. [Online; last accessed 27 March 2022].

[24] Utku Türk, Furkan Atmaca, Şaziye Betül Özateş, Gözde Berk, Seyyit Talha Bedir, Abdullatif Köksal, Balkız Öztürk Başaran, Tunga Güngör, and Arzucan Özgür. Resources for turkish dependency parsing: Introducing the boun treebank and the boat annotation tool. *Language Resources and Evaluation*, pages 1–49, 2021.

[25] Francis M. Tyers, Mariya Sheyanova, and Jonathan North Washington. UD Annotatrix: An annotation tool for Universal Dependencies. In *Proceedings of the 16th International Workshop on Treebanks and Linguistic Theories (TLT16)*, pages 10–17, 2018.

[26] Universal Dependencies. UD, 2022. URL https://universaldependencies.org. [Online; last accessed 2 May 2022].

[27] WebAnno. WebAnno - Documentation, 2022. URL https://webanno.github.io/webanno/documentation. [Online; last accessed 5 May 2022].

[28] Arife B. Yenice, Neslihan Cesur, Aslı Kuzgun, and Olcay Taner Yıldız. Introducing StarDust: A UD-based dependency annotation tool. In *Proceedings of the 16th Linguistic Annotation Workshop (LAW-XVI) within LREC2022*, pages 79–84, Marseille, France, June 2022. European Language Resources Association. URL https://aclanthology.org/2022.law-1.9.

A. BoAT-v1

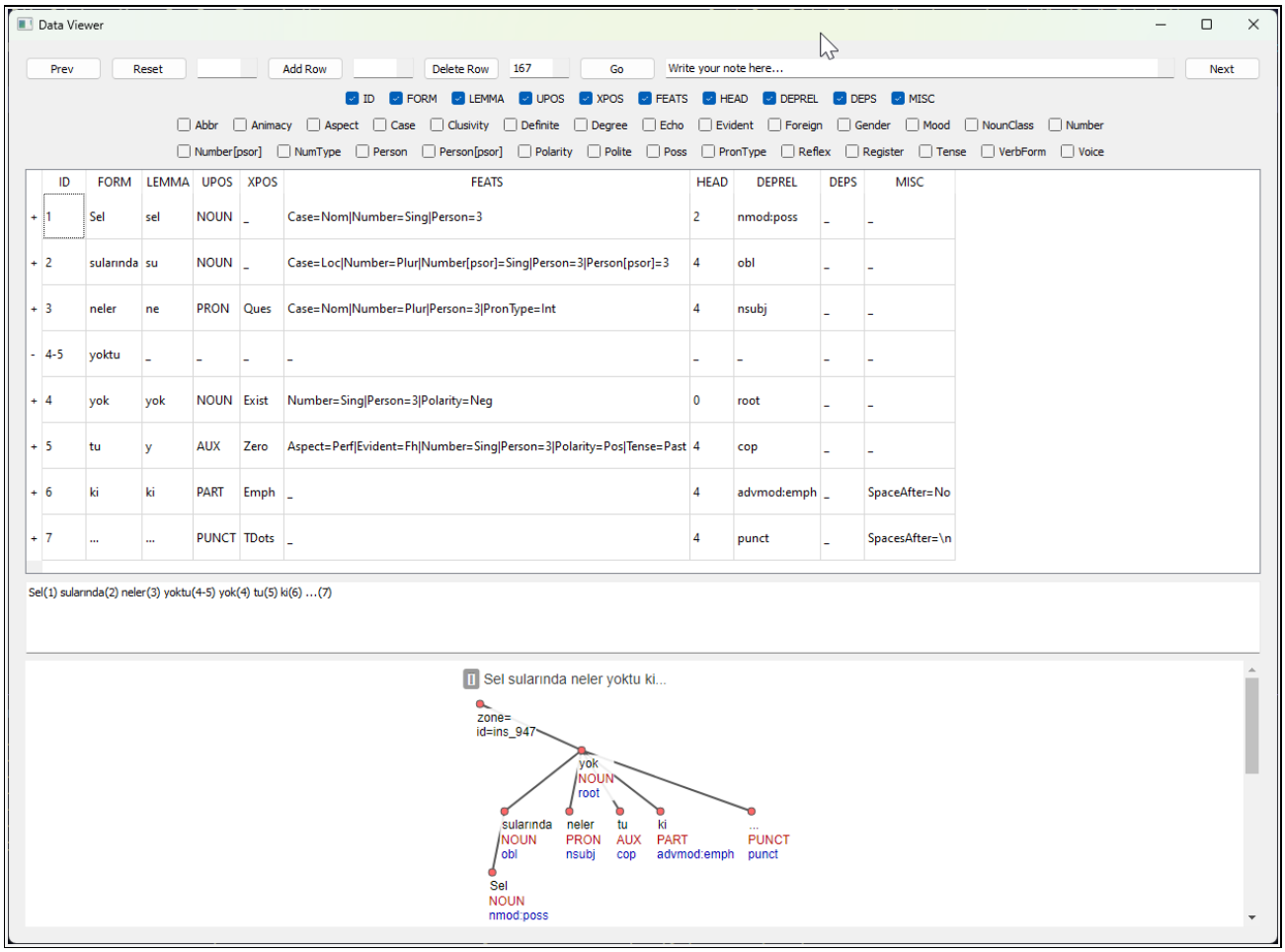

Figure 2. The annotation screen on BoAT-v1, captured while the sentence "Sel sularında neler yoktu ki..." is being annotated.