A Collaborative Web Tool for Linguistic Annotation

by

Salih Furkan Akkurt

Submitted to the Department of Computer
Engineering in partial fulfillment of
the requirements for the degree of
Bachelor of Science

Undergraduate Program in Computer Engineering
Boğaziçi University

Spring 2022

A Collaborative Web Tool for Linguistic Annotation

APPROVED BY:

Prof. Suzan Uskudarli .....
(Project Supervisor)

DATE OF APPROVAL: 6/12/2022

ACKNOWLEDGEMENTS

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

ABSTRACT

A Collaborative Web Tool for Linguistic Annotation

The value of quality treebanks is steadily increasing due to the crucial role they play in the development of natural language processing tools. The creation of such treebanks is enormously labor-intensive and time-consuming. Especially when the size of treebanks is considered, tools that support the annotation process are essential. Various annotation tools have been proposed, however, they are often not suitable for agglutinative languages such as Turkish. BoAT-v1 was developed for annotating dependency relations and was subsequently used to create the manually annotated BOUN Treebank (UD_Turkish-BOUN). In this work, we report on the design and implementation of a dependency annotation tool (BoAT-v2) based on the experiences gained from the use of BoAT-v1, which revealed several opportunities for improvement. BoAT-v2 is a multi-user and web-based dependency annotation tool that is designed with a focus on the annotator user experience to yield valid annotations. The main objectives of the tool are to: (1) support creating valid and consistent annotations with increased speed, (2) significantly improve the user experience of the annotator, (3) support collaboration among annotators, and (4) 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, and implementation of BoAT-v2 along with examples.

ÖZET

Dilbilimsel Anotasyon için Katılımcı Bir Web Aracı

Kaliteli ağaç yapılı derlemlerin değeri, doğal dil işleme araçlarının geliştirilmesinde oynadıkları rol sebebiyle artıyor. Bu derlemlerin yaratılması hem emek isteyen hem de zaman alan bir iş. Ağaç yapılı derlemlerin büyüklükleri göz önüne alındığında, etiketleme sürecini destekleyen araçların çok değerli olduğu anlaşılıyor. Çeşitli etiketleme araçları var, fakat bunlar sıklıkla Türkçe gibi eklemeli diller için uygun değiller. BOAT-v1 bağlılık ilişkilerini etiketlemek için geliştirilmişti ve ardından elle etiketlenen BOUN Treebank'i yaratmak için kullanıldı. Bu çalışmada, bir bağlılık ilişkilerini etiketleme aracının (BOAT-v2), geliştirme fırsatları ortaya çıkaran BOAT-v1'ın kullanımında kazanılan deneyimlere bağlı olarak, tasarım ve gerçeklemesini raporluyoruz. Bu yeni aracın baş hedefleri: (1) geçerli ve istikrarlı etiketlemelerin daha hızlı yapılabilmesini desteklemek, (2) etiketleyicilerin kullanıcı deneyimini epey geliştirmek, (3) etiketleyiciler arasındaki işbirliğini desteklemek, (4) bilimsel topluluklara fayda sağlamak için, açık kaynak ve kolaylıkla dağıtımı sağlanabilen web üzerinde kullanılabilen bir etiketleme aracı, esnek bir API ile birlikte, sağlamak. Bu rapor BOAT-v2'nun gereksinimlerini, tasarımını ve geliştirilmesini, yeni özellikleri vurgulayan etiketleme örnekleriyle detaylandırıyor.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ACRONYMS/ABBREVIATIONS

CoNLL	Computational Natural Language Learning
UD	Universal Dependencies
BoAT	Bogazici University Annotation Tool
DIP	Doğal Dil İşleme Platformu ("Natural Language Processing Platform")

1. INTRODUCTION AND MOTIVATION

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. Dependency annotation involves annotating each token of a sentence with linguistically relevant values. Universal Dependencies (UD) [1] is the most widely accepted standard for dependency annotations. The dependency annotation format of UD called CoNLL-U (Computational Natural Language Learning) defines a set of linguistic tags for annotation purposes. For example, Universal part-of-speech (UPOS) tag is used for annotating the part-of-speech of a token. The tag for morphological features (FEATS) is used for annotating 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 the complex morphology of such languages. Thus, the effort required to annotate agglutinative languages is 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, disrupting the flow.

BoAT-v1 [2] is an annotation tool that was developed to support dependency annotation of morphologically rich languages (MRLs) to produce treebanks compliant with the UD framework [1]. 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 dependency annotation tool (BOAT-v2) that has been designed based on the experience with BOAT-v1. In light of the feedback from the experience, we wanted the tool to be a web application that supports multiple users to enable a collaborative environment for annotations. 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 the annotator, (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 current prototype is being 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 will be made available on Boğaziçi University's NLP platform [3] and provided as an open-source resource.

The main contributions of this work are:

- Design of a dependency annotation tool based on requirements elicited from experienced annotators who are linguists,
- The development of a tool that takes into account the annotator 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 [4] and providing the code as an open-source resource.

The remainder of this report is organized as follows: Section 2 presents BOAT-v1, Section 3 presents related work, Section 4 describes the methods employed, Section 5 describes the requirements and design, Section 6 presents the implementation and new features, Section 7 presents a use case of annotation, Section 8 presents the results, and Section 9 provides a discussion along with our conclusions.

2. BoAT-v1

BoAT-v1 [5] is a standalone tool for annotating treebanks that is compatible with the UD framework [1] that is implemented using Python [6] and Qt [7]. It was specifically developed for annotating Turkish treebanks and is particularly suitable for agglutinative languages, however it can be used for other languages. It supports the annotation of a single treebank at a time. The 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 display errors. BoAT-v1 was used to create the BOUN Treebank [2, 8] – a manually annotated Turkish dependency treebank comprising close to 10 thousand sentences.

For each sentence to be annotated, the 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 [5]). The annotator manually enters values for each tag of each token. It supports the splitting and joining of lemmas which is particularly significant for agglutinative languages. Tokens that are split result in additional rows for each part. Figure B.2 in Appendix B.6 shows a sentence that is being annotated. 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 value of the FEATS tag often consists of numerous values. As such, they are difficult to read. BoAT-v1 supports viewing these features individually under their associated features (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.

3. STATE OF THE ART

Annotation tools may be characterized in terms of their accessibility and the support they provide for languages, various annotation categories, user interface modalities, standards, and multiple annotators. Adherence to standards is recommended to get the most benefit from the annotated data sources. Universal Dependencies [1] is an actively growing standard that intends to cover all languages and its support for agglutinative languages is evolving.

Several dependency annotation tools have been proposed such as *brat* [9], *UD Annotatrix* [10] and *DgAnnotator* [11, 12]. These tools are not developed for a specific language, but can be used with a variety of languages. They mostly rely on mouse-based user interaction, which is inefficient for annotating agglutinative languages due to the need for extensive annotation for most tokens.

The ITU Treebank Annotation Tool [13] was developed for Turkish. It was written in Java as an open-source standalone tool and has several versions. 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 [14].

WebAnno [15] 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 features of a token, it requires several mouse clicks which is impractical for MRLs. The sentences to be annotated are displayed consecutively, unlike many tools that focus on a single sentence at a time.

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 found to have in some spaces by the feedback of BoAT-v1.

4. METHODS

Numerous in-depth interviews with an active annotator have been conducted for requirements elicitation, requirements validation and prototype feedback. A Software Requirements Specification (SRS) document have been created as a result of these meetings and ample feedback. In these interviews, the annotator was asked questions about what the main cause of the loss of annotation flow was and how focus could be improved upon from BOAT-v1. The resultant requirements are displayed in the next section.

4.1. Project Management

(i) We have met every week for discussions related to the previous week's work.

(ii) We have also met at the DIP Meeting of our department on Fridays.

(iii) We communicated through Slack and Zoom.

(iv) We have used GitLab for the project repository and create issues there for everything, from bugs to documentation.

(v) The Wiki page on our GitLab repository includes documentation for requirements, implementation, meeting notes, etc.

5. SOFTWARE REQUIREMENTS

This section presents the software requirements elicited and validated through interviews with an annotator who is responsible for the creation of the BOUN Treebank.

5.1. Glossary for the Software Requirements Document

- **Annotation**: The task of parsing and identifying features of a text linguistically.
- **Autocomplete**: A shorthand writing system where a user can write the initials of a value and the system infers the value intended to be written.
- **Column**: A field of the forms chosen to be shown on the table by the annotator.
- **CoNLL-U**: A revised version of the CoNLL-X format[1] .
- **Dependency**: Specifically syntactical dependencies such that one individual form depends upon another.
- **Feature**: A kind of information of a certain word. A list can be found here[2] .
- **Field**: Parts of word lines. The possible fields are ID, FORM, LEMMA, UPOS, XPOS, FEATS, HEAD, DEPREL, DEPS, MISC and all the features individually. They are explained in detail here[3] .
- **Form**: A part of a sentence, whether it be a word or a punctuation symbol.
- **Help Page**: A web page explaining how to use the tool and possible values for features.
- **Index**: A form's order in the sentence (also called Word Index) or a sentence's order in the `conllu` file.
- **Lemma**: The base form of a set of words (e.g. break for the set {break, breaks, breaking, broke}).
- **Note**: Text written for a specific sentence that's to be used to explain the annotation choices.
- **Phrase**: A word or a few words that make sense together that may not make sense by themselves individually.

[1] <http://anthology.aclweb.org/W/W06/W06-2920.pdf>
[2] <https://universaldependencies.org/u/feat/index.html>
[3] <https://universaldependencies.org/format.html>

- **Row**: A form or lemma shown together with its features on the table.
- **Sentence**: One word or several words put together that can be divided into its lemmas to be annotated.
- **Table**: A rectangle divided by rows and columns showing the sentence being annotated, and its fields and forms in tabular form.
- **Tree**: A tree-like graph showing the dependencies annotated by the user for the current sentence.
- **Universal Dependencies**: A framework for annotating grammars of various natural languages. Also known as *UD*.
- **User**: A linguist annotating sentences through the tool.
- **Word**: A string of characters allowed for a certain natural language with no spaces that makes sense together with other words in a sentence.

5.2. Software Requirements Document (SRS)

5.2.1. Functional

1. User
    1. Accounts
        1. User shall be able to sign up or log in.
    2. Annotation
        1. Navigation
            1. User shall be able to select a sentence to be annotated.
            2. User shall be able to navigate to the preceding or succeeding sentences.
            3. User shall be able to navigate to a sentence by index.
            4. User shall be able to customize the annotation table by
                1. showing or hiding columns of the table.
                2. adding or deleting a row of the table, if the change is found to be valid.
        2. Information
            1. User shall be able to see the sentence being annotated at all times.

    2. User shall be able to see a dependency graph of a sentence.
    3. User shall be able to take notes during annotation.
  3. Modifications
    1. User shall be able to annotate a sentence by
      1. navigating to a specific sentence.
      2. modifying the categories and features corresponding to the fields of the CoNLL-U format in tabular format.
    2. User should be able to undo or redo the modifications as much as they want.
    3. User shall be able to reset the fields of the current sentence to the state they were in before the user opened the sentence.
3. Access
  1. User shall be able to access the application through a web browser.
  2. User shall be able to access a help page, related to annotating in general and keyboard shortcuts.
4. Search
  1. User should be able to search the `conllu` files or previous annotations by lemmas, phrases, dependencies & categories.
  2. User should be able to search using regular expressions, if chosen.

2. System
  1. Settings
    1. System should store settings & preferences of a user for use in a subsequent session.
    2. System shall store an annotator's sentences uploaded to the server & annotations done.
  2. Annotation
    1. System shall validate the annotations and check for errors in dependencies.
    2. System shall provide helpful error messages.
  3. Keyboard shortcuts
    1. System shall provide keyboard shortcuts for most tasks.

2. System shall provide autocompleted results for the fields being edited.

4. Database

    1. System shall provide a database to store user information.

5.2.2. Non-functional

1. Errors

    1. Errors given by the validation procedure shall comply with the CoNLL-U format of annotation of UD[4] .

[4] <https://universaldependencies.org/format.html>

6. IMPLEMENTATION

The annotation tool is implemented using Python [6], the web application development framework *Django* [16] and the API framework *Django REST Framework* (DRF) [17]. The webpages use Bootstrap [18]. 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 and compact horizontal to reduce the required screen real estate [20, 21].

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 the 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. 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.

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, a status is introduced in BOAT-v2 whose values may be: "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 5 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.

Improved searching for reference and consistency: A search functionality is introduced in BOAT-v2. Users are able to search for previously annotated sentences in combinations of surface text, UD tags, and features. Without a good search feature, an annotator would have to manually search the CONLL-U file for relevant cases (e.g. how to annotate some surface form’s UPOS tag). The surface form might have been inconsistently annotated, alas it is unlikely that a manual search would reveal this case. In such a case, the annotator would likely use the first encountered 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 text search would be hopeless as there would be too many hits since they occur very frequently.

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.

7. Using BoAT-v2

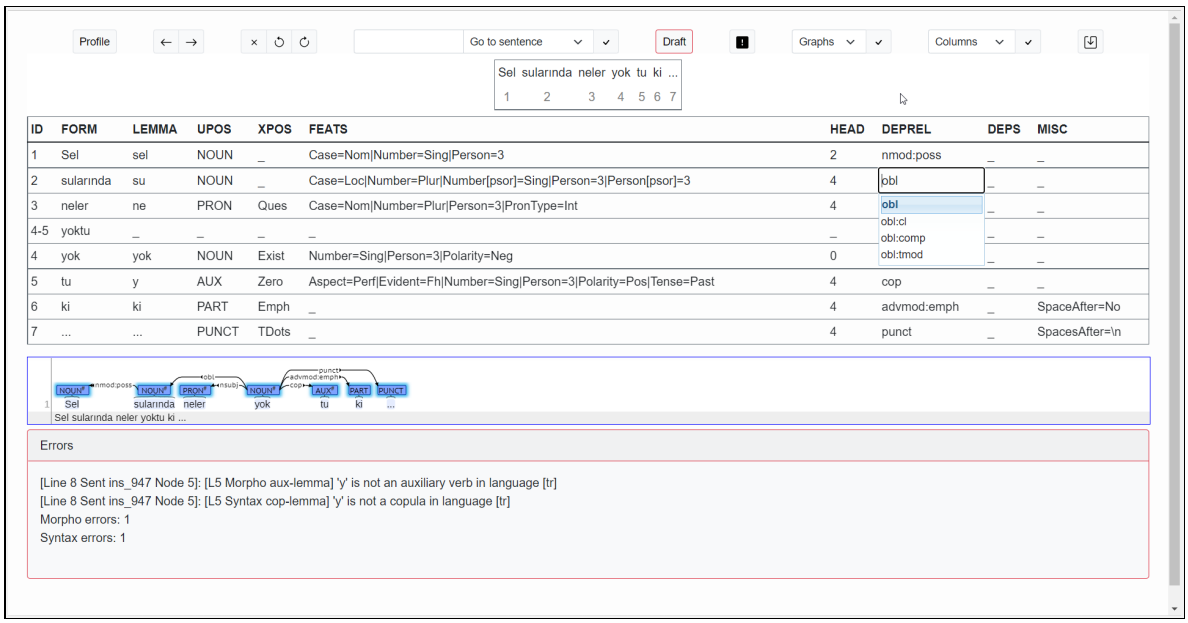

Figure 7.1. The annotation screen captured while the sentence "Sel sularında neler yoktu ki..." is being annotated. The DEPREL tag for the surface form "sularında" is being annotated by selecting among the valid alternatives that appear in the pop-up. Selections can be made with the use of arrow keys.

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 B.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.

8. RESULTS

We presented at The International Conference on Agglutinative Language Technologies as a Challenge of Natural Language Processing (ALTNLP) in Jun 2022, which was well-received, resulting in several downloads.

Currently, acceptance testing is being performed with annotators with linguistic background. Preliminary results are promising. The constant feedback-development cycle has had positive impact on the design, which is aimed to be user-oriented. A dockerized version of the tool is available on GitHub [22].

9. CONCLUSION AND DISCUSSION

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 6.

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 [4] has facilitated easy delivery and deployment. It will be made available on Boğaziçi University's NLP platform [3] as a demo as well 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.

This tool is in the testing phase and has had encouraging early feedback. 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 are encouraged by the early responses to 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 also resulted in significant speedup and improved user experience, which was reported as a qualitative observation by an annotator. Presently, our testing is focused on more extensive cases (annotation of sentences with varying degrees of complexity) and, importantly, the multi-user functionalities. For this purpose, we are in the process of recruiting several annotators with a background in linguistics.

10. FUTURE WORK

We are encouraged by the early responses to 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 also resulted in significant speedup and improved user experience, which was reported as a qualitative observation by an annotator. Presently, our testing is focused on more extensive cases (annotation of sentences with varying degrees of complexity) and, importantly, the multi-user functionalities. For this purpose, we are in the process of recruiting several annotators with a background in linguistics.

This summer (2022), we will be pursuing these points:

(i) Endangered languages: The Linguistics Department of our university has been contacted to pursue using the tool for an endangered language.
(ii) Uzbek: Researchers at ALTNLP2022 has shown interest in using the tool to create a treebank for the Uzbek language.
(iii) Semi-automated methods to annotate treebanks partially
(iv) A user manual

REFERENCES

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

2. Türk, U., F. Atmaca, Ş. B. Özateş, G. Berk, S. T. Bedir, A. Köksal, B. Ö. Başaran, T. Güngör and A. Özgür, "Resources for Turkish dependency parsing: Introducing the BOUN treebank and the BoAT annotation tool", *Language Resources and Evaluation*, pp. 1–49, 2021.

3. Boğaziçi University, "TABILAB Repository Home", , 2022, https://nlp.cmpe.boun.edu.tr, [Online; last accessed 24 April 2022].

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

5. Türk, U., F. Atmaca, Ş. B. Özateş, A. Köksal, B. Ozturk Basaran, T. Gungor and A. Özgür, "Turkish Treebanking: Unifying and Constructing Efforts", *Proceedings of the 13th Linguistic Annotation Workshop*, pp. 166–177, Association for Computational Linguistics, Florence, Italy, Aug. 2019, https://www.aclweb.org/anthology/W19-4019.

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

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

8. Türk, U., F. 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, https://universaldependencies.org/tr, [Online; last accessed 27 March 2022].

9. Stenetorp, P., S. Pyysalo, G. Topić, T. Ohta, S. Ananiadou and J. Tsujii, "brat: a Web-based Tool for NLP-Assisted Text Annotation", *Proceedings of the Demonstrations Session at EACL 2012*, Association for Computational Linguistics, Avignon, France, April 2012.

10. Tyers, F. M., M. Sheyanova and J. N. Washington, "UD Annotatrix: An annotation tool for Universal Dependencies", *Proceedings of the 16th International Workshop on Treebanks and Linguistic Theories (TLT16)*, pp. 10–17, 2018.

11. Attardi, G., "DgAnnotator", , 2022, `http://medialab.di.unipi.it/Project/QA/Parser/DgAnnotator/`, [Online; last accessed 27 March 2022].

12. Universal Dependencies, "UD tools", , 2022, `https://universaldependencies.org/tools`, [Online; last accessed 22 April 2022].

13. Pamay, T., U. Sulubacak, D. Torunoğlu-Selamet and G. Eryiğit, "The Annotation Process of the ITU Web Treebank", *Proceedings of The 9th Linguistic Annotation Workshop*, pp. 95–101, Association for Computational Linguistics, Denver, Colorado, USA, Jun. 2015, `https://aclanthology.org/W15-1610`.

14. Torunoğlu-Selamet, D., T. Pamay, U. Sulubacak and G. Eryiğit, "The Annotation Process of the ITU Web Treebank", , 01 2015.

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

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

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

18. Bootstrap, "Bootstrap v5.1 Documentation", , 2022, `https://getbootstrap.`

com/docs/5.1, [Online; last accessed 5 May 2022].

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

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

21. Pyysalo, S., "conllu.js", , 2022, http://spyysalo.github.io/conllu.js, [Online; last accessed 25 March 2022].

22. Furkan Akkurt, "BoAT v2", , 2022, https://github.com/furkanakkurt1335/boat/, [Online; last accessed 12 Jun 2022].

APPENDIX A: DATA AVAILABILITY STATEMENT

The annotation tool has been made available as open-source on GitHub. It has been dockerized to make it accessible and easy to deploy.

APPENDIX B: BoAT v2 - A Web-Based Dependency Annotation Tool with Focus on Agglutinative Languages

Parts of the paper, presented at ALTNLP2022: The International Conference on Agglutinative Language Technologies as a Challenge of Natural Language Processing, are included here.

B.1. Related Work

Annotation tools may be characterized in terms of their accessibility and the support they provide for languages, various annotation categories, user interface modalities, standards, and multiple annotators. Adherence to standards is recommended to get the most benefit from the annotated data sources. Universal Dependencies [1] is an actively growing standard that intends to cover all languages and its support for agglutinative languages is evolving.

Several dependency annotation tools have been proposed such as *brat* [9], *UD Annotatrix* [10] and *DgAnnotator* [11, 12]. These tools are not developed for a specific language, but can be used with a variety of languages. They mostly rely on mouse-based user interaction, which is inefficient for annotating agglutinative languages due to the need for extensive annotation for most tokens.

The ITU Treebank Annotation Tool [13] was developed for Turkish. It was written in Java as an open-source standalone tool and has several versions. 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 [14].

WebAnno [15] 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 features of a token, it requires several mouse clicks which is impractical for MRLs. The sentences to be annotated are displayed consecutively, unlike many tools that focus on a single sentence at a time.

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 found to have in some spaces by the feedback of BOAT-v1.

B.2. 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 size of treebanks necessitates that they are annotated via 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: While annotating a sentence, annotators may 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 features of the FEATS tag, and surface forms. This requirement reduces the cognitive load as well as supports consistency among annotations.

Keyboard-oriented input: Annotations of agglutinative languages require annotation of several features for individual tokens 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 is frustrating. Keyboard-based interaction must be offered for all possible tasks.

Support for Sentence annotation: To support the speed 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 is important for agglutinative languages, which requires splitting of lemmas. Words in MRLs tend to have morphemes stacked on roots. Annotators must be able to refine/correct automatically parsed entries, which for agglutinative languages includes *splitting of lemmas*.

Use of screen real estate and customization: Annotators annotate numerous sentences every 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, each annotator has their unique preferences for how they annotate a sentence. 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 convey 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 [4] to support its accessibility.

B.3. Implementation

The annotation tool is implemented using Python [6], the web application development framework *Django* [16] and the API framework *Django REST Framework* (DRF) [17]. The webpages use Bootstrap [18]. 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 and compact horizontal to reduce the required screen real estate [20, 21].

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 the 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. 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.

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, a status is introduced in BoAT-v2 whose values may be: "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 B.2 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.

Improved searching for reference and consistency: A search functionality is introduced in BOAT-v2. Users are able to search for previously annotated sentences in combinations of surface text, UD tags, and features. Without a good search feature, an annotator would have to manually search the CONLL-U file for relevant cases (e.g. how to annotate some surface form's UPOS tag). The surface form might have been inconsistently annotated, alas it is unlikely that a manual search would reveal this case. In such a case, the annotator would likely use the first encountered 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 text search would be hopeless as there would be too many hits since they occur very frequently.

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.

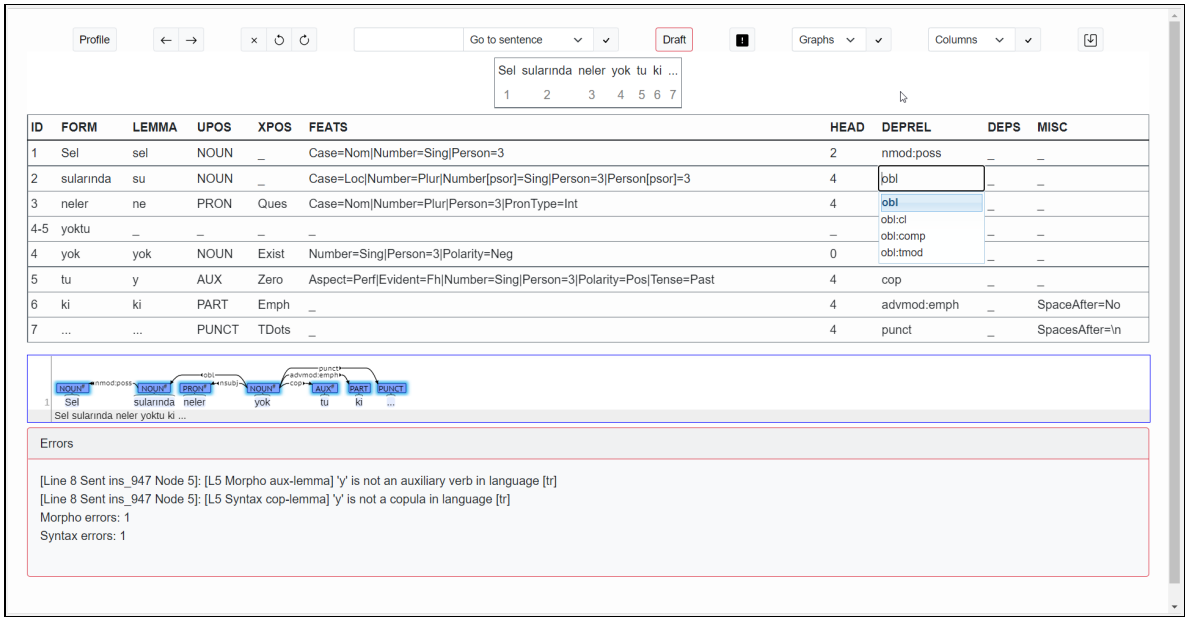

Figure B.1. The annotation screen captured while the sentence "Sel sularında neler yoktu ki..." is being annotated. The DEPREL tag for the surface form "sularında" is being annotated by selecting among the valid alternatives that appear in the pop-up. Selections can be made with the use of arrow keys.

B.4. Using BoAT-v2

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 B.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.

B.5. 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 B.3.

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 [4] has facilitated easy delivery and deployment. It will be made available on Boğaziçi University's NLP platform [3] as a demo as well 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.

This tool is in the testing phase and has had encouraging early feedback. 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 are encouraged by the early responses to this tool and anticipate its exten-

sions. In fact, our implementation of BoAT-v2 resulted in a revision request for BoAT-v1 to include the focus enhancing features. This also resulted in significant speedup and improved user experience, which was reported as a qualitative observation by an annotator. Presently, our testing is focused on more extensive cases (annotation of sentences with varying degrees of complexity) and, importantly, the multi-user functionalities. For this purpose, we are in the process of recruiting several annotators with a background in linguistics.

B.6. BoAT-v1

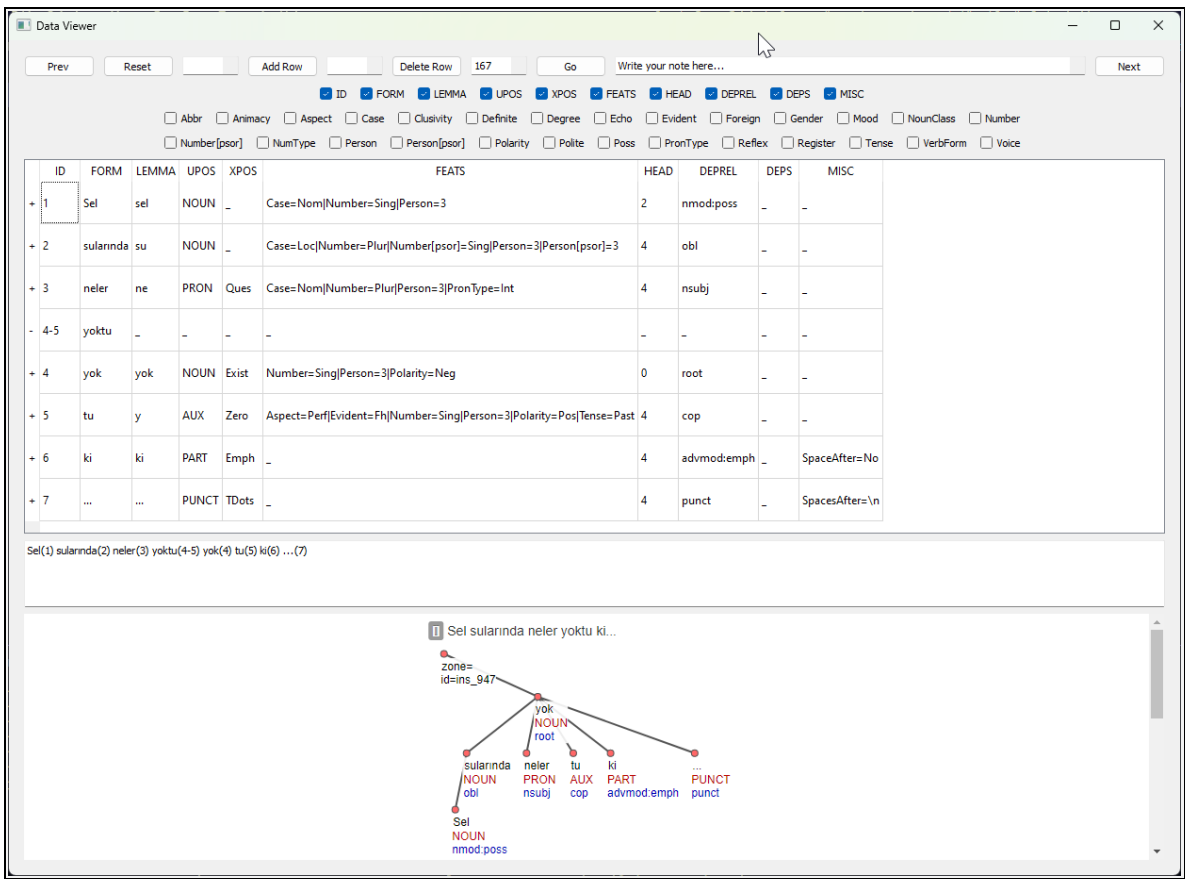

Figure B.2. The annotation screen on BoAT-v1, captured while the sentence "Sel sularında neler yoktu ki..." is being annotated.