A Dual-Interface Syntax Tree Editor with Automated Assessment for Learning Management Systems

Atanas Atanasov
Faculty of Slavic Studies,
Sofia University "St. Kliment Ohridski"
atanasov@slav.uni-sofia.bg

Abstract

While syntax trees are fundamental to computational linguistics, existing visualization tools often lack research flexibility or native Learning Management System (LMS) integration. This paper presents a dual-interface syntax tree editor consisting of a standalone web application for linguistic annotation[1] and a Moodle question type for educational use. The system provides an interactive drag-and-drop interface for constructing hierarchical syntactic structures directly in the browser.

The Moodle integration features an automated grading engine based on a weighted Leaf-Ancestor metric. Unlike PARSEVAL-based approaches, this method evaluates hierarchical accuracy by comparing bottom-up syntactic paths via token-based weighted Levenshtein distance. The grading logic treats the root, parts-of-speech, and internal hierarchy as separate components, distinguishing between structural omissions and terminological errors. This enables nuanced partial credit and formative feedback consistent with educational assessment principles. The system supports scalable assessment and linguistic annotation, with particular relevance for less-resourced languages such as Bulgarian.

Keywords: Syntax Tree Editor, Automated Assessment, Computer-Aided Learning, Moodle question types, Weighted Leaf-Ancestor metric.

1 Introduction

Syntax trees are a fundamental component of both computational linguistics research and syntax education. For linguistic researchers, constructing accurate syntactic hierarchies is an essential task for analyzing sentence structures, creating treebanks, and producing syntactic annotations (Kiss and Alexiadou, 2015). In the educational context, the active construction of these diagrams helps students transition from explicit grammatical rule comprehension to implicit syntactic knowledge, allowing them to visualize complex dependencies (Van Valin, 2004; Kroeger, 2005). However, balancing the flexibility required for advanced research annotation with the scalability needed for educational assessment remains an open problem.

Several software tools have been developed to facilitate syntax tree drawing, yet they often address only partial aspects of the educational and research workflows. Standalone desktop applications like the original Syntax Tree Editor and TreeForm (Derrick and Archambault, 2010) offer interactive drag-and-drop interfaces and produce high-quality vector graphics suitable for scientific publication. While highly useful for linguists, such tools lack native integration into Learning Management Systems (LMS), requiring students to export images and instructors to manually grade them. Conversely, web-based generators such as RSyntaxTree (Hasebe, 2009) produce static images from bracketed text input, lacking both the intuitive graphical manipulation desired by researchers and the automated assessment capabilities required by educators. Table 1 summarizes the feature comparison between existing popular solutions and the proposed system.

To properly contextualize the proposed tool within the current landscape of syntactic technology, it is essential to distinguish between

[1] https://syntree.ezik.bg/

automated deep-learning syntactic parsers and interactive, human-in-the-loop tree editors. While neural parsers execute automated statistical sequence labeling, visual environments like TreeForm (Derrick and Archambault, 2010), RSyntaxTree (Hasebe, 2009), and the suite of manual annotation tools maintained within the Universal Dependencies (UD) framework – such as browser-based editors like UD Annotatrix, ConlluEditor, and ArboratorGrew – serve a fundamentally different purpose. These graphical environments are explicitly designed to facilitate human-centric structural validation, educational diagramming, and the manual curation of gold-standard treebanks, which are themselves required to train and evaluate deep-learning parsing architectures. While many modern UD-aligned platforms are specifically optimized for word-level dependency relations or CoNLL-U format modification, our system provides a unique dual-interface setup tailored specifically for phrase-structure constituency tree manipulation with native LMS integration for academic assessment, establishing an accessible pipeline for student training and pedagogical dataset generation.

Addressing the disparity between flexible linguistic research and scalable educational assessment, this paper presents a novel, dual-interface syntax tree editor. The system comprises a standalone web application designed for efficient linguistic annotation by scientists, and a native Moodle question type tailored for educational deployment. The editor provides an interactive, browser-based drag-and-drop interface that allows users to construct and modify complex hierarchical structures dynamically, without requiring external software installations or complex local setups.

A key feature of the Moodle integration is the introduction of an automated grading engine that evaluates student submissions using an advanced Isolated Root Weighted Leaf-Ancestor metric. By analyzing bottom-up syntactic paths and decoupling the evaluation of the root, parts-of-speech, and internal hierarchy, the algorithm overcomes the strict penalization and over-grading problems typical of standard PARSEVAL metrics. This enables nuanced partial credit and targeted formative feedback consistent with modern educational principles. Consequently, this system provides a scalable solution that serves both as an advanced annotation tool for linguists and a computer-aided learning application for educators, making it highly applicable to multi-lingual contexts, including less-resourced languages such as Bulgarian.

2 System Architecture and Design

To satisfy the dual requirements of flexible linguistic research and scalable educational assessment, the syntax tree editor is engineered as a decoupled, client-side TypeScript application. It operates in two deployment environments: a standalone web application for linguistic annotation and a custom Moodle question type plugin (qtype_syntree) for educational integration.

2.1 Rendering Engine and User Interface

Unlike older desktop tools requiring external runtimes (e.g., Java), the editor operates entirely in-browser using Scalable Vector Graphics (SVG). This guarantees resolution-independent rendering, making the output highly suitable for interactive manipulation across screen sizes and high-quality vector exports for scientific publications.

User interaction is driven by the unified Pointer Events API, providing seamless drag-and-drop capabilities across desktop (mouse) and mobile (touch) devices. One specific technical challenge in creating readable syntax trees is maintaining the symmetric visual alignment of nodes, particularly when pairing bold syntactic labels (e.g., NP, VP) with regular-weight lexical items. To resolve this, the system's inline node editor utilizes an off-screen HTML5 Canvas API (measureText()) to calculate the exact pixel width of the input strings dynamically. Furthermore, the layout engine employs a two-pass algorithm and a Depth-First Search (DFS) traversal to order the leaf nodes, which mathematically guarantees a crossing-free layout for the branches, preserving the formal properties of constituency trees.

In addition to standard constituency trees, the rendering engine supports the visualization of syntactic movement through trace arrows. These are implemented as complex SVG path elements with an automatic collision avoidance algorithm, which dynamically calculates orthogonal routing to prevent intersections with existing nodes. Furthermore, to support academic publishing, the system features an advanced export module that provides direct SVG downloads or renders the vector-based DOM into high-resolution PNGs via a scaled Canvas.

2.2 Moodle LMS Integration

In the educational deployment, the editor is tightly integrated into the Moodle ecosystem as a custom question type. The integration utilizes Moodle's Asynchronous Module Definition (AMD) architecture to load the compiled JavaScript bundle dynamically within the quiz interface.

Instead of relying on a separate backend server or complex API calls for state management, the plugin employs a robust real-time serialization approach. As the student constructs or modifies the tree via the graphical interface, the core engine serializes the hierarchical structure into a standard linguistic bracket notation (e.g., [S [NP [D The] [N boy]] [VP [V ran]]]). This serialized string is continuously synchronized with a hidden HTML input field within the Moodle quiz form. Upon submission, Moodle's native form processing captures the bracket notation string, which is subsequently parsed by the backend PHP grading engine. This architecture ensures high interoperability, strict compliance with institutional data security policies, and seamless data flow between the graphical frontend and the automated assessment backend.

A visual demonstration of the instructor's question configuration interface and the corresponding student interactive environment within the Moodle LMS ecosystem is provided in Appendix A (Figure 1 and Figure 2).

3 Automated Assessment via a Weighted Leaf-Ancestor Metric

Standard metrics for evaluating syntax trees, such as PARSEVAL, are widely used in computational linguistics but present significant drawbacks in educational contexts. They often over-grade "flat" trees that lack hierarchical depth and strictly penalize minor terminological errors, failing to provide the nuanced formative feedback required for learning. To address this, the proposed Moodle integration employs a custom grading engine based on an advanced Isolated Root Weighted Leaf-Ancestor metric.

The grading engine evaluates student submissions against an instructor-defined reference tree by decoupling the assessment into three independent components: Part-of-Speech (POS) accuracy (15%), Root identification (5%), and Internal Hierarchy (80%). This separation ensures pedagogical fairness by preventing a "domino effect," where a single error at the highest node disproportionately degrades the score of an otherwise correct structural derivation. To accommodate diverse educational goals, the system allows instructors to dynamically customize and override these default weights within the Moodle interface, realigning the automated scoring with specific course levels or targeted learning outcomes.

The core innovation of the assessment engine lies in the Internal Hierarchy evaluation. For each lexical item in the sentence, the system extracts a bottom-up syntactic path from the leaf to the root. To isolate purely structural knowledge, the leaf node and the pre-terminal POS tag are filtered out from this path. The system then compares the student's syntactic path to the reference path using a Weighted Levenshtein Distance algorithm. While severe structural errors – such as omitted or extraneous hierarchical levels (insertions and deletions) – incur a full penalty cost of 1.0, terminological errors (substitutions, e.g., mislabeling a Noun Phrase as a Verb Phrase) incur a reduced cost of 0.66.

For a given student path S and reference path R, the structural score per word is calculated using mathematical normalization:

$$Score_{word} = \max\left(0{,}1 - \frac{\mathrm{dist}_{weighted}(S,R)}{\max(\mathrm{len}(S),\mathrm{len}(R))}\right)$$

Furthermore, the engine incorporates strict pedagogical safeguards. If a student submits a completely flat structure without internal hierarchy, the hierarchy score automatically defaults to zero, preventing unearned credit. By differentiating between fundamental structural omissions and surface-level labeling mistakes, this algorithm provides scalable, fair partial credit consistent with modern assessment principles.

3.1 Evaluation of the Grading Engine

Standard evaluation metrics like PARSEVAL present dual pedagogical failures: they over-penalize minor terminological label mistakes, yet often over-grade completely "flat" trees that lack hierarchical depth. To demonstrate how the proposed Isolated Root Weighted Leaf-Ancestor metric mathematically resolves these issues, consider the reference sentence evaluated against an instructor-defined gold standard: [S [NP [D The] [N boy]] [VP [V ran]]].

If a student submits an incorrect structure that mislabels the Noun Phrase as a Verb Phrase ([S [VP [D The] [N boy]] [VP [V ran]]]), the grading engine isolates the structural tracking. For the token "boy", the bottom-up reference path (excluding the POS tag) is [NP, S], while the student's layout yields [VP, S]. The Weighted Levenshtein Distance algorithm detects a single substitution operation. According to the system design, while structural insertions or deletions incur a full penalty of 1.0, terminological substitutions incur a reduced cost of 0.66 to reflect partial student knowledge. Applying the system's normalization formula:

This mathematically guarantees that the student receives a partial credit of 67% for the token by establishing the correct hierarchical depth and grouping, precisely isolating and penalizing only the surface-level labeling error without triggering a catastrophic scoring failure for the entire phrase.

For a given student path S and reference path R, the structural score per word is calculated using mathematical normalization:

$$Score_{word} = \max\left(0{,}1 - \frac{0.66}{\max(2{,}2)}\right) = 0.67$$

Conversely, if a student submits a completely flat tree (attaching all terminals directly to the root with no intermediate hierarchy), the algorithm detects the complete absence of internal paths. Because the reference path length for "boy" is 2, and the student's path is empty (length 0), the deletion distance is evaluated as 2.0 (with a full 1.0 penalty per deletion).

The maximum function safeguard ensures that the hierarchy score correctly defaults to zero, rigorously penalizing the fundamental structural omission.

The practical implementation within the Slavic Philology cohort at Sofia University fully substantiated these algorithmic expectations. Students adapted to the web-based editor seamlessly, demonstrating an intuitive grasp of the interactive drag-and-drop interface without requiring prior review of the detailed user documentation. In independent study settings, the instantaneous feedback provided by the automated scoring engine served as a vital formative resource during self-preparation. For instructors, the framework drastically minimized the administrative time required for manual grading and script evaluation. Furthermore, deploying the tool during synchronous lectures enabled the interactive analysis of a significantly larger volume of syntactic structures compared to previous semesters, which relied on traditional paper-based workflows or the legacy standalone Syntax Tree Editor.

4 Educational and Research Applications

The proposed dual-interface system addresses distinct but complementary use cases in both linguistic research and higher education.

In the research context, the standalone web application serves as an accessible and highly efficient annotation tool for linguists. The construction of syntactic resources, such as treebanks, is particularly critical for the computational advancement of less-resourced languages like Bulgarian. It is important to specify that while contemporary Bulgarian NLP is well-supported by robust processing pipelines for foundational tasks like morphological tagging and basic parsing, it remains relatively less-resourced regarding open-access, web-integrated pedagogical platforms and collaborative tools for interactive treebank validation. The standalone interface specifically targets this infrastructural gap. The interactive browser-based interface, coupled with the ability to export resolution-independent Scalable Vector Graphics (SVG), directly supports the creation of publication-ready linguistic analyses and the rapid expansion of natural language datasets without requiring complex local software setups.

In the educational context, the Moodle-integrated question type addresses the pedagogical challenges of teaching syntax to large student cohorts by shifting the focus toward active structural modeling. This approach aligns with broader Computer-Aided Language Learning (CALL) frameworks, where interactive, software-driven tree diagramming has been shown to significantly enhance students' implicit syntactic competence (El-Garawany, 2025). By embedding this functionality natively within an LMS, the system facilitates scalable, computer-aided learning. Instructors can deploy complex syntactic assignments efficiently, while the automated grading engine provides immediate, formative feedback. This automated approach not only dramatically reduces the administrative and grading burden on educators but also ensures a standardized, objective evaluation of students' structural and terminological knowledge.

5 Conclusion and Future Work

This paper presented a novel, dual-interface syntax tree editor designed to bridge the gap between flexible linguistic annotation and scalable computer-aided education. By operating entirely within the browser using resolution-independent SVG rendering, the standalone application provides a robust environment for researchers constructing treebanks for less-resourced languages. Simultaneously, its native integration into Moodle as a custom question type addresses the pedagogical need for scalable assessment in computational linguistics courses. The implementation of the Isolated Root Weighted Leaf-Ancestor metric represents a significant advancement over standard PARSEVAL approaches, enabling nuanced partial credit and formative feedback that are crucial for student development.

Future development will focus on expanding the system's analytical capabilities. A primary objective is the integration of the Universal Dependencies (UD) framework to support dependency parsing visualization alongside the current constituency trees. Additionally, future work will explore incorporating AI-assisted annotation, where backend parsing models generate preliminary tree structures that researchers or students can subsequently refine via the drag-and-drop interface. Finally, a large-scale empirical evaluation across multiple higher education institutions is planned to quantitatively measure the system's impact on students' syntactic competence and knowledge retention.

References

Derrick, D. and Archambault, D. (2010). TreeForm: Explaining and exploring grammar through syntax trees. *Literary and Linguistic Computing*, 25(1): 53–66.

El-Garawany, M. S. M. (2025). Using TreeForm for Enhancing English Language Majors' EFL Syntactic Competence. *rEFLecTions*, 32(2): 754–786.

Hasebe, Y. (2009). *RSyntaxTree* [Software]. Available at: https://yohasebe.com/rsyntaxtree/ (Accessed: 14 April 2026).

Kiss, T. and Alexiadou, A. (2015). *Syntax - Theory and Analysis: An International Handbook* (Vol. 3). De Gruyter Mouton.

Kroeger, P. R. (2005). *Analyzing Grammar: An Introduction*. Cambridge University Press.

Van Valin, R. D. (2004). *An Introduction to Syntax*. Cambridge University Press.

Appendix A User Interfaces

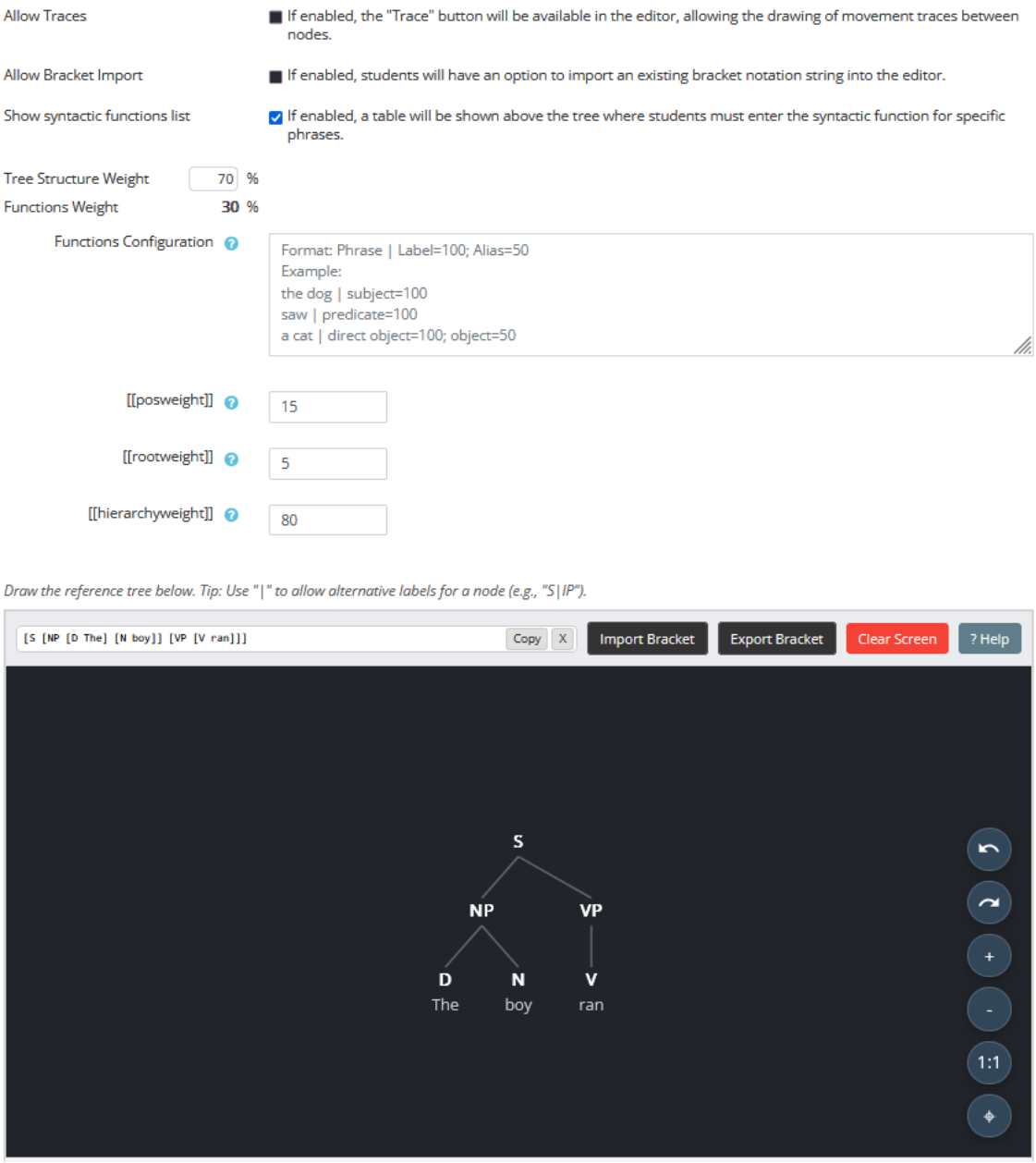

Figure 1: The instructor's question design and configuration interface within the Moodle platform, displaying reference bracket inputs and scoring weight parameters.

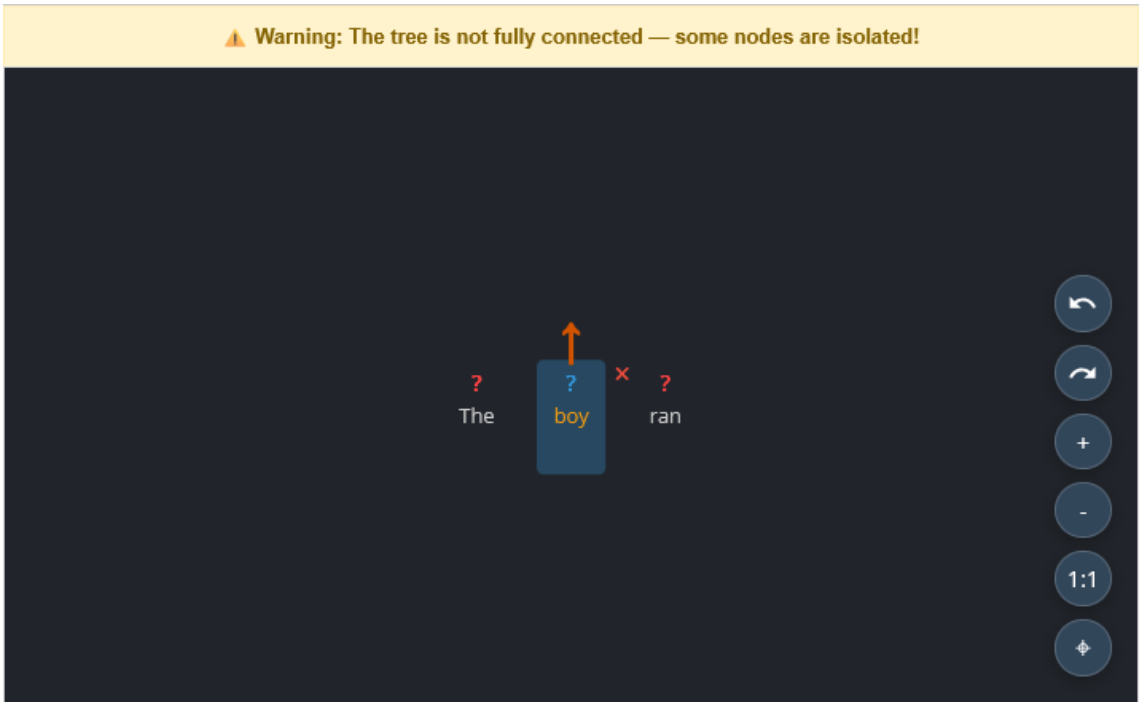

Figure 2: The student interactive environment within Moodle, featuring the responsive drag-and-drop canvas for structural syntax tree construction.

0