A GitHub-based Workflow for Annotated Resource Development

Brandon Waldon **Nathan Schneider**
Georgetown University
{bw686, nathan.schneider}@georgetown.edu

Abstract

Computational linguists have long recognized the value of version control systems such as Git (and related platforms, e.g., GitHub) when it comes to managing and distributing computer code. However, the benefits of version control remain under-explored for a central activity within computational linguistics: the development of annotated natural language resources. We argue that researchers can employ version control practices to make development workflows more transparent, efficient, consistent, and participatory. We report a proof-of-concept, GitHub-based solution which facilitated the creation of a legal English treebank.

1 Introduction

Linguistic annotation is an important pillar of the empirical enterprise that supports modern computational linguistics. A recent review notes that "corpus resources... remain highly relevant for testing and studying [NLP] systems" (Opitz et al., 2025: 4), even as these resources take a less central role in system training. By augmenting corpus data with high-quality annotations, "people skilled at language analysis can ensure meaningful evaluation of NLP systems" (ibid).

However, creating a valuable annotated dataset is time-consuming and labor-intensive, and some common practices can undermine the usefulness and quality of the end result. For example, behind each "gold" annotation may be several non-trivial analytical decisions reached through careful adjudication. Unfortunately, researchers tend not to make, or publicly share, detailed records of these processes. As a result of low project **transparency**, dataset users may have no way of determining the original justification for a given annotation.

Moreover, linguistic annotation practices tend to vary widely in terms of the assistive tools made available to annotators. Providing annotators with access to tools that automatically visualize and/or validate annotations can facilitate more **efficient** and more **consistent** (i.e., less error-prone) resource development (Bontcheva et al., 2010; Stenetorp et al., 2012). However, there are high overhead costs for creating such tools from scratch, meaning that less mature annotation projects are often pursued with more primitive annotation technologies.

Finally, not all workflows permit the kinds of robust community **participation** that help to sustain linguistic annotation projects over time. Though most projects are sustained primarily by the efforts of a core development team, outside researchers can make valuable contributions by identifying annotation errors or adding new annotations. To make full use of these non-core contributors, it is desirable to develop resources on platforms that facilitate open communication between a project's core developers and the broader research community.

We argue that researchers can address these issues with resource development workflows that employ version control systems (such as Git) and online services for interacting with such systems (such as GitHub). Though computational linguists have long recognized the value of version control for managing and distributing computer code, we demonstrate that version control systems and services also serve to make linguistic annotation procedures more **transparent**, **efficient**, **consistent**, and **participatory**.

In what follows, we recap the core principles behind version control generally and Git/GitHub in particular. We then present our GitHub-based annotation workflow in general form. Next, we report a proof-of-concept implementation, which facilitated the creation of a treebank of legal English.

2 Version control and Git/GitHub

In this section, we briefly review the concept of a version control system (VCS) and the core principles underlying Git/GitHub, with a focus on prop-

Proceedings of the 19th Linguistic Annotation Workshop (LAW-XIX-2025), pages 326–331

erties that facilitate our proposed workflow.

A VCS records changes to a file repository over time, allowing teams to track modifications, compare versions, and revert to previous states when needed. VCS adoption enables developers to create and modify files while maintaining a complete project history within the repository.

Git is a widely employed VCS. A Git **branch** is a parallel instance of the repository with a change history that may diverge from that of the central version of the project (as reflected by the 'main' branch). Branches allow project contributors to develop new features or fixes without affecting the main codebase before the changes are ready to be integrated. A Git **commit** records the changes made to repository files at a specific point in time. Each commit contains a unique hash identifier and includes a message describing the changes made. Commits create a traceable history of modifications, allowing viewers to understand when and why particular changes were implemented.

GitHub is a web-based hosting service for managing and sharing Git projects. While Git provides the foundational version control capabilities, GitHub extends these with a social platform that enables web-based collaboration. On GitHub, **pull requests** enable developers to propose changes from their working branch to the main branch. Pull requests serve as a collaborative space where team members can review file changes, provide feedback, and discuss modifications before changes are merged from a working branch to the main branch. GitHub **actions** specify automated procedures triggered by repository events (such as commits or pull requests). Actions serve to automate repetitive tasks such as testing code or writing files.

3 Application to linguistic annotation

Notably, GitHub has already proven to be valuable for large-scale linguistic annotation projects such as Universal Dependencies (de Marneffe et al., 2021), which employs GitHub as a forum for discussing annotation guidelines and as a tool for maintaining existing datasets.[1] Our proposed workflow goes a step further by integrating GitHub directly at the resource development stage. This level of integration results in a comprehensive record of annotation decisions (and annotator discussions) for each individual annotation in the dataset.

This workflow (Figure 1) starts with two conceptual roles performed by project participants: the *annotator* role and the *manager* role.[2] The manager organizes the annotation project by populating a subdirectory of the repository with "stub" entries. These entries include pre-annotated text, possibly with some pre-processing (e.g., tokenization). These entries, and/or their associated filenames, may also include project-relevant metadata.

From the main GitHub branch where stub entries reside, the annotator creates a working branch.[3] Within this working branch, the annotator completes stub entries, adding annotations according to the project guidelines. Each time an annotator commits changes to their working branch, two GitHub actions are automatically triggered: a visualization action and a validation action. The visualization action creates a graphical representation of the annotated data and commits it to the annotator's branch. The validation action triggers a script that heuristically verifies that the annotation conforms to conventions of the annotation schema.

When an annotator completes their annotations, they initiate a pull request to merge their changes back into the main branch. The manager reviews the pull request. This review is facilitated by the action-generated graphical representation, which enables the manager to inspect the proposed contribution without having to manually read through the raw text of the annotation file. The manager and annotator can also review the output of the validation action to ensure the annotation is well-formed.

The manager and annotator can discuss the proposed contributions by leaving comments on the pull request. Ultimately, the manager has two options: approve the changes and merge them into the main branch, or request additional edits from the annotator. In the latter case, the annotator makes edits on the annotator branch and then requests a subsequent review from the manager.

Upon successful merging of annotated entries into the main branch, a statistics action is auto-

[1] https://github.com/universaldependencies

[2] A single individual may perform multiple roles, and the tasks of a single role may be delegated to multiple individuals.

[3] Because the manager adds stub files directly to the main branch, that branch will consist of both incomplete and complete files until all annotations are merged. This creates minor inconveniences for data browsing and statistics collection. On an alternative implementation, the manager is tasked with creating each stub file on a dedicated branch, immediately opening a draft pull request assigned to the annotator. This modified approach would maintain a cleaner main branch containing only completed annotations; it would also eliminate the need for external assignment tracking.

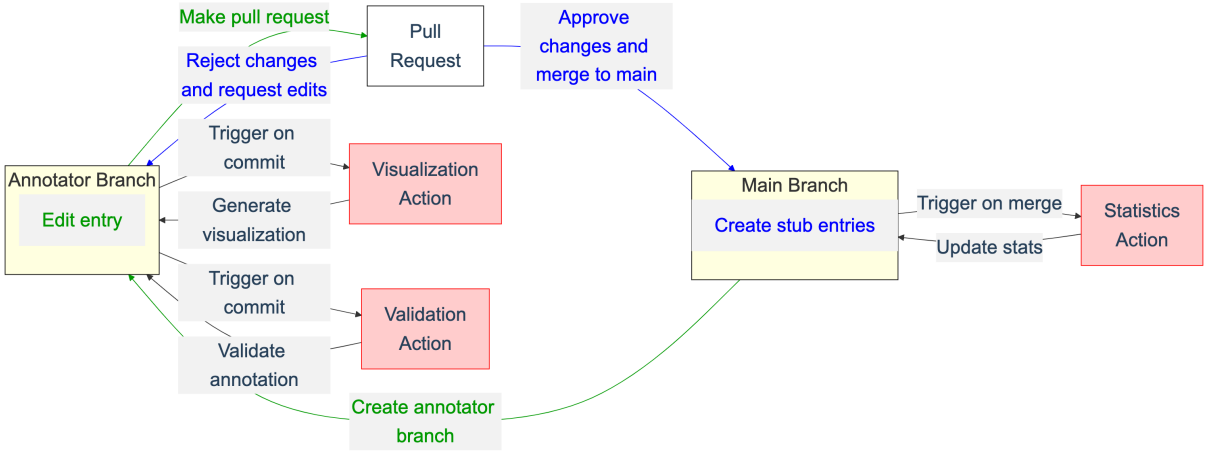

Figure 1: Workflow schema. Blue text indicates *manager* tasks; green text indicates *annotator* tasks.

matically triggered. This process updates project statistics, which may include information about overall project progress or summary statistics of the annotations themselves.

In what follows, we show that this workflow can be implemented in a way that promotes the four values presented in Section 1: **transparency**, **consistency**, **efficiency** and community **participation**.

4 Demonstration: treebanking

We applied this workflow while developing a treebank of legal US English in CGELBank (Reynolds et al., 2023), a treebanking formalism that extends the descriptive theory of English syntax presented in the Cambridge Grammar of the English Language (CGEL, Huddleston and Pullum, 2002).

The core team consisted of five researchers. Each team member performed the tasks of the annotator role, while the tasks of the manager role were performed primarily by the two senior members of the team. One member working in the manager role populated the main branch with stub files in the project-native `.cgel` data format (Figure 2; see Reynolds et al. 2023, Sec. 5 for more discussion), with each file corresponding to one sentence of the treebank. In addition to the raw sentence text and other relevant metadata, each stub file contained an automated tokenization of the sentence.

The annotated sentences were sourced from US federal statutes as compiled in the US Code by the Office of the Law Revision Counsel (OLRC) of the US House of Representatives.[4] The OLRC publishes the US Code in XML format according to a standardized schema known as United States Legislative Markup (USLM). Each sentence of the

```
# sent_id = ...
# text = the Attorney General
# sent = the Attorney General
(NP
    :Det (DP
        :Head (D :t "the"))
    :Head (Nom
        :Head (N :t "Attorney")
        :Mod (AdjP
            :Head (Adj :t "General"))))
```

Figure 2: Example of the `.cgel` data format, illustrating analysis of the noun phrase *the attorney general*.

treebank is associated with an ID derived from unique USLM metadata associated with the parent element of the sentence. For ease of browsing and cross-referencing the treebank data, we found it helpful to designate a short unique prefix to each sentence ID, e.g. `usc-039` for sentence 39.

For each sentence, the assigned annotator created a new working branch from the main branch of the project's GitHub-hosted repository. The annotator then manually corrected the automated tokenization and added lemma and part-of-speech tags according to CGELBank conventions (Reynolds et al., 2024). Tree editing was facilitated by ActiveDOP (van Cranenburgh, 2018), a browser-based graphical treebanking tool which utilizes an active learning parser (disco-dop, van Cranenburgh et al. 2016). To enable editing of `.cgel`-format trees, we extended a CGELBank-customized version of ActiveDOP reported by Reynolds et al. (2023). Once the annotator was finished using the tool, they exported the `.cgel`-format tree from ActiveDOP and appended it to the corresponding stub file. The an-

[4] `https://uscode.house.gov/`

notator then saved and committed their file changes to their working branch.

Some annotators opted to interface with Git from the command line (and subsequently 'push' their commits to the project's GitHub repository), while others utilized GitHub's built-in text editor user interface to edit and commit changes directly from their web browser. Once the annotator's changes had been committed to their working branch on GitHub, a visualization action automatically generated a LaTeX rendering of the `.cgel`-format tree as a `.pdf` file and committed that file to the working branch. A second validation action verified that the tree did not have any obvious errors.

The annotator then opened a pull request on the main branch. Another team member, assuming the manager role, reviewed the pull request by inspecting the changed files. The LaTeX rendering provided the reviewer with a convenient, easy-to-read graphical representation of the user's annotation. The reviewer and annotator could discuss the annotation through comments left on the pull request. In the event that the reviewer requested changes, the annotator could modify the relevant `.cgel` file, which automatically re-triggered the visualization action to update the LaTeX `.pdf` of the tree. This procedure is partly illustrated in Figure 3.

Once the reviewer approved the annotation and merged it to the main branch, an automatically-triggered action generated summary statistics of the treebank, including counts of lexical nodes and category/function labels, average tree depth, and a list of high-frequency lemmas.

5 Discussion

Our project repository[5] is not simply a static collection of gold annotations; the repository's commit history and pull request comments also form a dynamic public record of the decision-making processes that led to that gold data. This feature of our development workflow enhances project **transparency**, providing future dataset users with a means of determining how we adjudicated hard cases of linguistic analysis.

As a new treebanking formalism with a relatively small research community, CGELBank lacks the breadth of specialized annotation tools enjoyed by more established projects, e.g., Universal Dependencies (de Marneffe et al., 2021).[6] We used

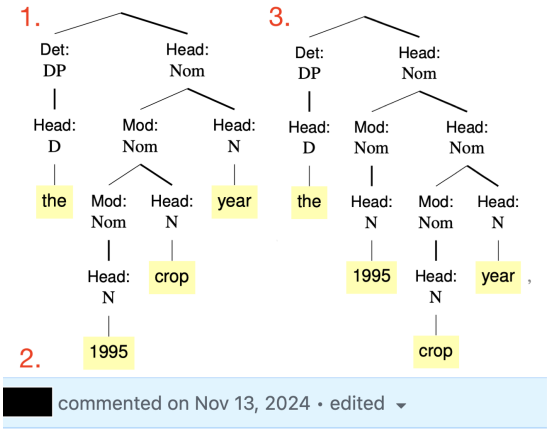

Figure 3: (1): excerpt of a GitHub action-generated LaTeX visualization for an annotator's CGELBank tree annotation; (2): excerpt of a reviewer comment on the pull request containing the annotation; (3): the visualization action is re-triggered after the annotator commits their edits, yielding a modified LaTeX rendition.

GitHub actions – relatively simple scripts which execute in a GitHub repository – to deliver some of the functionality of standalone annotation tools (i.e., automated visualization and validation), in addition to using and extending a bespoke CGELBank annotation tool. We used these actions in a way that allowed the annotator and reviewer to **efficiently** discuss and adjudicate a proposed annotation. These actions – especially the automated valiation – also promote **consistency** by enabling annotators and reviewers to quickly spot errors.

Lastly, the public nature of GitHub strongly encourages community **participation**. Anyone with a GitHub account can comment on the project by posting a GitHub *issue* (a discussion thread used to track project-related matters). The broader community can also create pull requests to suggest corrections to the dataset (or to add new data).

6 Related work

To a limited extent, previous work has discussed the utility of version control for developing annotated linguistic resources. Palmer and Xue (2010) recommend that annotators employ a VCS protocol to promote data security and integrity as a resource is developed. San (2016) implements a Git-based procedure to develop a dataset of phonetic transcriptions for three indigenous Australian languages. On this procedure, annotators' contributions are tracked through Git commits, and Git "hooks" (automated scripts) automatically re-

[5] https://github.com/nert-nlp/legal-cgel/
[6] https://universaldependencies.org/tools.html

compute corpus statistics upon merge. Our proposed workflow builds on this approach by leveraging the social functionality of GitHub to facilitate adjudication, foster community participation, and create a persistent open record of the design and analysis choices that shape the final corpus product.

Previous work has also explored the value of VCS technologies for maintaining previously-developed resources. Rosenberg (2012) and Steiner (2017) discuss how version control could help research communities record (and disseminate) changes and corrections to speech corpus annotations. Dumitru et al. (2024) design and implement a VCS for managing *dynamic* speech corpora of the kind envisioned by Rosenberg.

Previous work has focused largely on applying VCS protocols in the context of annotated speech corpora. To our knowledge, we report the first application of a VCS-based workflow to syntactic treebanking. However, as discussed in Section 3, GitHub already plays a significant role in the ongoing maintenance of the Universal Dependencies project, including as a forum for discussing errors and updates to annotation conventions.

7 Limitations

Though our workflow offers several advantages for linguistic annotation, we have not presented a quantitative comparison of annotation speed or accuracy against alternative workflows. Additionally, while GitHub actions provide useful automation, developing and maintaining custom validation and visualization scripts requires a non-trivial number of technical prerequisites, including familiarity with the YAML-based workflow syntax associated with GitHub actions. Finally, annotators unfamiliar with version control in general (or Git in particular) may face a learning curve associated with the core concepts of Git repository management.

8 Conclusion

We presented a GitHub-based workflow for linguistic annotation. We provided a proof-of-concept implementation of this workflow for syntactic treebanking, demonstrating that this workflow promotes four values that enhance the usefulness and quality of annotated linguistic resources. Future work could extend this approach to other types of linguistic annotation tasks beyond treebanking, such as semantic role labeling or discourse analysis. Moreover, the workflow could be adapted to support multiple independent annotations followed by adjudication, leveraging Git's branching model to manage parallel annotation efforts.

Finally, there are opportunities to integrate GitHub with external annotation tools through the GitHub Apps framework,[7] which enables third-party software to directly perform common GitHub operations such as writing commits, opening/commenting on pull requests, and triggering automated workflows. In ongoing work, we are extending such functionality to ActiveDOP (van Cranenburgh, 2018), the tree editor employed in our CGELBank treebanking demonstration, so that annotators can participate in a GitHub-based workflow without leaving the annotation environment.

Computational linguistics continues to depend on high-quality linguistic annotation to support empirically-informed natural language analysis and data-driven system development. By embracing version control practices and technologies, we can foster more rigorous, collaborative, and sustainable approaches to this essential practice.

9 Acknowledgments

We thank Micaela Wells, Devika Tiwari, and Meru Gopalan, who participated as annotators for the CGELBank project described in the paper. We additionally thank Meru Gopalan for his assistance in developing the visualization action script described in Section 4. We also thank Brett Reynolds for providing helpful feedback on many annotations. Finally, we gratefully acknowledge the insightful comments of three anonymous LAW reviewers. This research was supported in part by NSF award IIS-2144881 (to NS).

References

Kalina Bontcheva, Hamish Cunningham, Ian Roberts, and Valentin Tablan. 2010. Web-based collaborative corpus annotation: Requirements and a framework implementation. In *New Challenges for NLP Frameworks (NLPFrameworks 2010)*, pages 20–27, Valletta, Malta. ELRA Language Resources Association.

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

Vlad Dumitru, Matthias Boehm, Martin Hagmüller, and Barbara Schuppler. 2024. Version control for

[7] https://docs.github.com/en/apps/overview

speech corpora. In *Proceedings of the 20th Conference on Natural Language Processing (KONVENS 2024)*, pages 303–308, Vienna, Austria. Association for Computational Linguistics.

Rodney Huddleston and Geoffrey K. Pullum, editors. 2002. *The Cambridge Grammar of the English Language*. Cambridge University Press, Cambridge, UK.

Juri Opitz, Shira Wein, and Nathan Schneider. 2025. Natural language processing RELIES on linguistics. *Computational Linguistics*. To appear.

Martha Palmer and Nianwen Xue. 2010. *Linguistic Annotation*, chapter 10. John Wiley & Sons, Ltd.

Brett Reynolds, Aryaman Arora, and Nathan Schneider. 2023. Unified syntactic annotation of English in the CGEL framework. In *Proc. of LAW*, pages 220–234, Toronto, Canada.

Brett Reynolds, Nathan Schneider, and Aryaman Arora. 2024. CGELBank annotation manual v1.1. *Preprint*, arXiv:2305.17347.

Andrew Rosenberg. 2012. Rethinking the corpus: moving towards dynamic linguistic resources. In *Proceedings of Interspeech 2012*, pages 1392–1395, Portland, USA. International Speech Communication Association.

Nay San. 2016. Using version control to facilitate a reproducible and collaborative workflow in acoustic phonetics. In *Proceedings of the Sixteenth Australasian International Conference on Speech Science and Technology (SST2016)*, pages 341–344, Parramatta, Australia. Australasian Speech Science and Technology Association.

Ingmar Steiner. 2017. A devops manifesto for speech corpus management. In *Proceedings of the 28th Conference on Electronic Speech Signal Processing (ESSV)*, pages 160–166, Saarbrücken, Germany. Deutsches Forschungszentrum für Künstliche Intelligenz.

Pontus Stenetorp, Sampo Pyysalo, Goran Topić, Tomoko Ohta, Sophia Ananiadou, and Jun'ichi Tsujii. 2012. brat: a web-based tool for NLP-assisted text annotation. In *Proceedings of the Demonstrations at the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 102–107, Avignon, France. Association for Computational Linguistics.

Andreas van Cranenburgh. 2018. Active DOP: A constituency treebank annotation tool with online learning. In *Proceedings of the 27th International Conference on Computational Linguistics: System Demonstrations*, pages 38–42, Santa Fe, New Mexico. Association for Computational Linguistics.

Andreas van Cranenburgh, Remko Scha, and Rens Bod. 2016. Data-oriented parsing with discontinuous constituents and function tags. *Journal of Language Modelling*, 4(1):57–111.