SoftwareX 26 (2024) 101755

Contents lists available at ScienceDirect

SoftwareX

journal homepage: www.elsevier.com/locate/softx

Original software publication

Discovering and exploring cases of educational source code plagiarism with Dolos

Rien Maertens *, Maarten Van Neyghem, Maxiem Geldhof, Charlotte Van Petegem, Niko Strijbol, Peter Dawyndt, Bart Mesuere

Department of Applied Mathematics, Computer Science and Statistics, Ghent University, Ghent, Belgium

ARTICLE INFO

Keywords:
Web app
Plagiarism
Source code
Academic dishonesty
Cheating
Learning analytics
Educational data mining
Online learning
Programming language

ABSTRACT

Source code plagiarism is a significant issue in educational practice, and educators need user-friendly tools to cope with such academic dishonesty. This article introduces the latest version of Dolos, a state-of-the-art ecosystem of tools for detecting and preventing plagiarism in educational source code. In this new version, the primary focus has been on enhancing the user experience. Educators can now run the entire plagiarism detection pipeline from a new web app in their browser, eliminating the need for any installation or configuration. Completely redesigned analytics dashboards provide an instant assessment of whether a collection of source files contains suspected cases of plagiarism and how widespread plagiarism is within the collection. The dashboards support hierarchically structured navigation to facilitate zooming in and out of suspect cases. Clusters are an essential new component of the dashboard design, reflecting the observation that plagiarism can occur among larger groups of students. To meet various user needs, the Dolos software stack for source code plagiarism detection now includes a self-hostable web app, a JSON application programming interface (API), a command line interface (CLI), a JavaScript library and a preconfigured Docker container. Clear documentation and a free-to-use instance of the web app can be found at https://dolos.ugent.be. The source code is also available on GitHub.

Code metadata

Current code version	v2.6.0
Permanent link to code/repository used for this code version	https://github.com/ElsevierSoftwareX/SOFTX-D-24-00117
Permanent link to Reproducible Capsule	n/a
Legal Code License	MIT
Code versioning system used	git
Software code languages, tools, and services used	TypeScript, JavaScript, Ruby, Rails, Docker, Vue.js
Compilation requirements, operating environments & dependencies	NodeJS, Yarn, Python 3, GCC
If available Link to developer documentation/manual	http://dolos.ugent.be/docs
Support email for questions	dodona@ugent.be

1. Motivation and significance

The rise in computer science enrolments [1] and the inclusion of computational thinking and software development in secondary and higher education curricula [2,3] has resulted in an increase in source code production for classroom assignments. This worldwide trend comes with its own set of challenges, including source code plagiarism [4,5]. Cosma and Joy [6] define this phenomenon as *"Source-code plagiarism in programming assignments can occur when a student reuses source-code authored by someone else and, intentionally or unintentionally, fails to acknowledge it adequately, thus submitting it as his/her own work.* [...]". The temptation for students to circumvent learning and to cheat on assessments increases with higher stakes and access to online sources, peer-to-peer communication and generative AI [7–9].

The migration from paper-based to digital computer science education has increased the use of software tools for detecting source code plagiarism. These tools aid educators in detecting, proving, and preventing such forms of educational dishonesty by automating the

* Corresponding author.
E-mail address: rien.maertens@ugent.be (Rien Maertens).

https://doi.org/10.1016/j.softx.2024.101755
Received 22 February 2024; Received in revised form 26 April 2024; Accepted 29 April 2024
Available online 9 May 2024

process of finding, comparing, and visualising similar code fragments among large collections of source files. However, a number of studies on source code plagiarism rely on unpublished tools that are not or no longer publicly available [10]. Additionally, the process of downloading, installing and running plagiarism detection tools can become tedious and error-prone, which negatively impacts the user experience. Through our interactions with more than 1000 secondary and higher educational institutions [11], we have repeatedly heard that this is a significant barrier for a large group of educators, especially for those with varying levels of technical expertise. Proper interpretation of the results is also a bottleneck [12]. This might explain why many educators still refrain from using source code plagiarism detection [13,14]. MOSS[1] [15] and JPlag[2] [16] are currently the most popular free-to-use tools for plagiarism detection in educational source code. However, MOSS is a closed-source, externally hosted web service that requires running a local submission script, and JPlag requires local software installation to perform the detection.

For natural language processing, the significance of plagiarism detection is emphasised by the abundance of commercial and free web apps available [17,18]. Some of these tools are specifically designed for educational purposes. They enable educators to conduct plagiarism detection checks directly from their browser, without the need for complex installation procedures or multiple tools. For source code, various commercial web apps for plagiarism detection exist, such as Codequiry,[3] Copyleaks.[4] and Gradescope[5] However, none of these apps are fully open-source and free to use.

To fill this gap, we expanded on the initial prototype of Dolos [19]. This initial version was already competitive with state-of-the-art tools in terms of performance and prediction accuracy while using a language-agnostic pipeline [19]. However, it also needed local installation and its user interface was quite basic. The latest major release of Dolos (version 2.x) addresses these issues and offers numerous improvements. A new web app has been developed that obviates the need for local installation, with a free-to-use instance hosted at https://dolos.ugent.be. We also provide easy configuration and documentation on how to self-host the Dolos web app. This allows educators to comply with local privacy policies or to use its API for seamless integration into online learning environments. While the command line interface (CLI) from the first version is still supported, all new and improved features are now also accessible from the new web app. The web app also supports sharing reports in an easy, secure, and privacy-preserving way, stimulating discussions among educators about possible source code plagiarism. In addition, the web interface has been redesigned to include new powerful dashboards that allow educators to zoom in from the entire collection, over clusters and pairs, to individual source files. All visualisations have been significantly improved for better responsiveness and the plagiarism detection pipeline has been optimised for faster runtimes and reduced memory footprints. Finally, the user experience has also been improved with faster load times, support for anonymisation, automatic programming language detection, highlighting differences between two source files, sharing online dashboards safely with colleagues, and a new packaging strategy for programming language support.

2. Illustrative example

This section provides instructions on how to use the Dolos web app to detect plagiarism in a collection of programming assignment submissions. From an educator's perspective, the process involves two steps: (i) uploading source files and (ii) checking dashboards for suspected cases. To follow along, it is recommended to use the free-to-use instance hosted at Ghent University (https://dolos.ugent.be) with either your own collection of submissions or our sample dataset. You can also take a video tour at https://dolos.ugent.be/tour.

For more information on how to install and run the CLI locally, self-host a local instance of the web app using Docker, or use the JavaScript library directly, please refer to the online documentation at https://dolos.ugent.be/docs.

Step 1: data submission

The web app's launchpad consists of two panels (Fig. 1). The left panel features an upload form for submitting new collections of source files, while the right panel contains a searchable table for retrieving previously submitted collections and reviewing their analysis results.

To submit a new collection, begin by selecting a ZIP archive containing the source files from the local file system. The app can automatically detect the programming language, or it can be manually specified by selecting from a drop-down list. The archive may also contain a CSV-formatted file with metadata provided by online learning environments such as submission timestamps, authors and free-form labels. Dolos currently supports the simple CSV-format exported by Dodona [11].

Upon submission, the app launches a server-side job that executes the source code similarity analysis pipeline. Jobs usually finish within a few seconds, and the results are then accessible for further examination. The analysis is intentionally finely tuned for smaller source code files. For collections containing more than 1000 files, files with over 1000 lines of code, or when integrating Dolos into an automated pipeline, we recommend using the Dolos CLI or Javascript library. These restrictions are not hardcoded into the web app, but the pipeline will gracefully abort when there is insufficient memory to complete the analysis.

Each uploaded collection of source and metadata files is stored server-side, along with the analysis results. Instead of relying on user accounts, each result is assigned a unique **secret key** that is stored in the browser's local storage. The table in the launchpad allows for easy access to previous analysis results, which can be shared with colleagues or deleted both client and server-side.

Step 2: exploring analysis results

Like other plagiarism detection tools, Dolos' server-side analysis pipeline merely automates the detection of highly similar code fragments shared between source files and calculates pairwise similarities between each pair of files in the collection. However, gathering enough convincing evidence is undoubtedly the most challenging aspect of dealing with educational source code plagiarism. This task is challenging to fully automate, but the web app assists the educator's expert eye with new and carefully crafted **plagiarism analytics dashboards**.

Dolos provides dashboards for various subsets of source files in the collection: the complete **collection**, a **cluster** of files, a **pair** of files, and a **single** file. This creates a hierarchical structure of linked dashboards at different zoom levels. Moving between linked dashboards provides a natural zooming experience when investigating suspected cases of plagiarism.

The exploration of the complete collection starts at an **overview dashboard** (Fig. 2). Its analytics and visualisations provide an immediate impression of whether the collection contains suspected cases of plagiarism and the extent of plagiarism within the collection. Clues can be found, for example, by contrasting the highest and average pairwise similarities between files, relating source files to their nearest neighbour in terms of global similarity (both available as a histogram and a list), and inspecting the number and size of file clusters.

The same underlying goal led us to visualise the hierarchically structured subsets of the collection as a **plagiarism graph** (Fig. 3). The graph shows suspect files (as nodes coloured by label), pairs (as edges

[1] https://theory.stanford.edu/~aiken/moss/
[2] https://github.com/jplag/JPlag
[3] https://codequiry.com/
[4] https://copyleaks.com/code-plagiarism-checker
[5] https://www.gradescope.com/

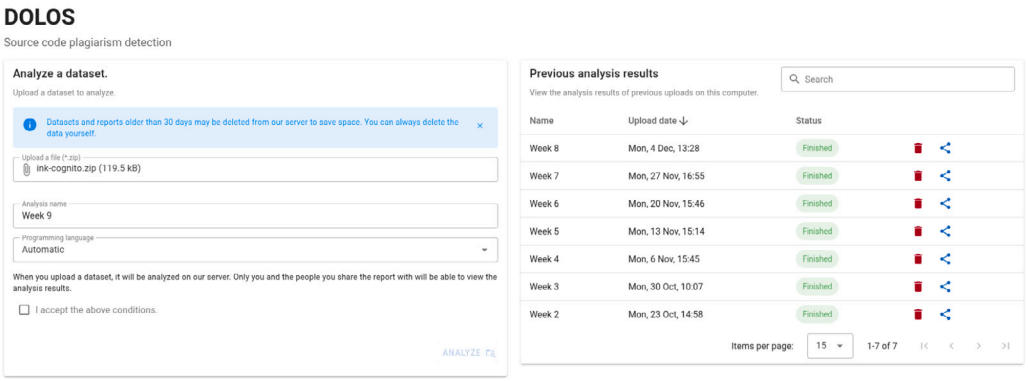

Fig. 1. Launchpad of the Dolos web app. Left panel: upload form for submitting a new collection of source files. Right panel: searchable table for accessing, deleting and sharing previously submitted collections.

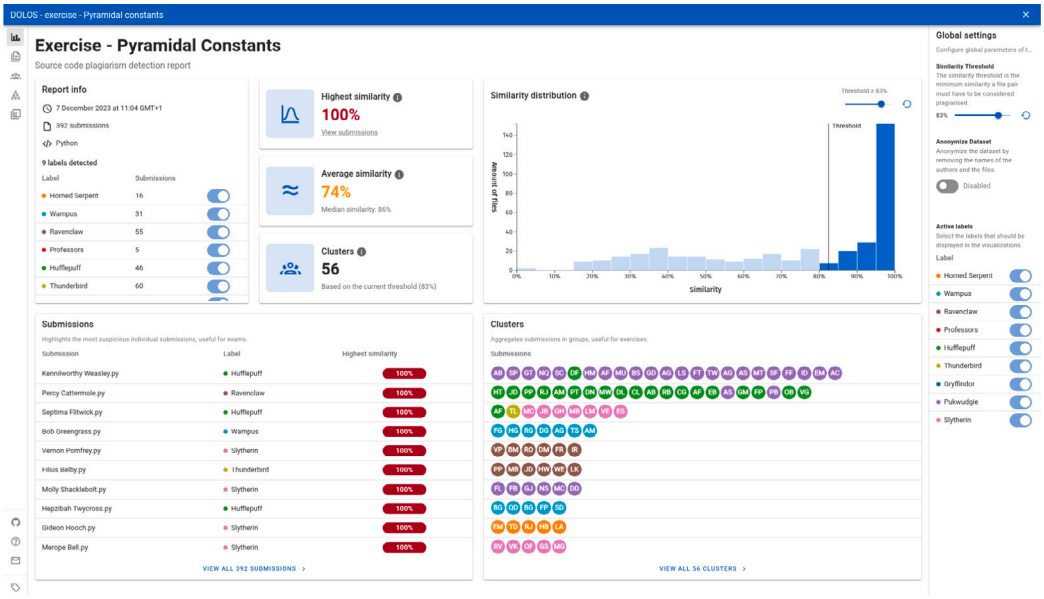

Fig. 2. The overview dashboard's analytics and visualisations summarise the plagiarism detection results. This specific report suggests that plagiarism is prevalent in this publicly available collection of source files.

drawn between nodes whose global similarity exceeds a similarity threshold), and clusters (coloured regions that group nodes connected by edges).

Existing source code plagiarism tools traditionally only report potential plagiarism from the perspective of file pairs. Larger groups of collaborating students quickly result in an unmanageable list of file pairs (e.g. 10 students result in 45 file pairs), which may be scattered across a list of reported file pairs. However, seeing the same data visualised as a clustered graph feels very intuitive. As a result, the **cluster** concept is now an integral part of the Dolos dashboard design as a separate hierarchical level. This feature helps distinguish between peer-to-peer plagiarism events (two students sharing code) and broadcast events (larger groups of students sharing code, e.g. via social media). The cluster dashboard reconstructs the distribution timeline based on submission timestamps. This feature is useful for tracking the original author or observing how the distribution process has evolved over time.

The **pair dashboard** displays two source files side by side (Fig. 4). It assists educators in identifying adequate and conclusive evidence that high global similarity or lengthy shared fragments are not coincidental. Students intentionally employ various obfuscation techniques to conceal that they have copied someone else's code [10]. Both plagiarism detection pipelines and educators must try to see through this. The pair dashboard offers two views: one highlights matching fragments found by the Dolos plagiarism detection pipeline, while the other shows a diff [20] of the two submissions. When educators land on a pair dashboard, the app automatically selects the most relevant view for the two source files at hand.

All dashboards share three global settings, which can be modified in a dedicated panel (expanded from the far right of the top navigation bar; Fig. 2) or in some panels within the dashboards themselves. Suspect pairs and clusters are delineated based on a global **similarity threshold**. Dolos employs a simple heuristic to automatically determine an appropriate initial value for this threshold. All analytics and visualisations can be **anonymised** to present dashboards in a privacy-friendly manner. Discussing the impact and consequences of plagiarism with students could be part of a preventive strategy [19,21]. **Label-based filtering** is used to control which subset of the total collection is considered by the dashboards.

3. Software description

All Dolos source code is available in a public monorepo on GitHub.[6] and in the Zenodo software repository [22] This section describes the

[6] https://github.com/dodona-edu/dolos

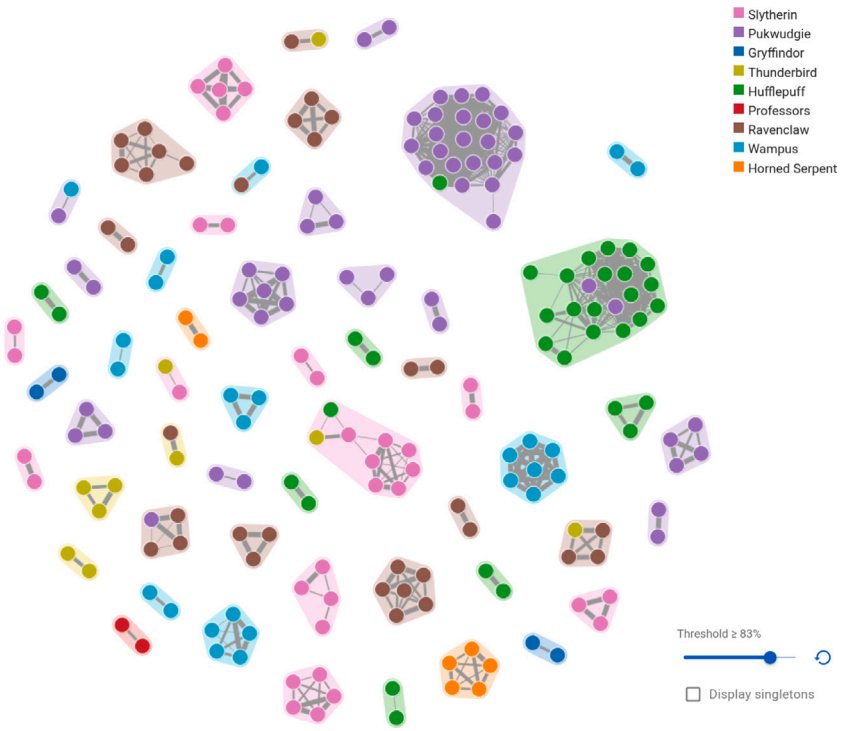

Fig. 3. Graph showing suspected cases of plagiarism within the same collection of source files used for Fig. 2. Each node represents a submission and has a colour corresponding to its labels. Edges connect nodes whose pairwise similarity exceeds an adjustable threshold (bottom right). Unconnected submissions are currently hidden, but can be shown by checking the "Display singletons" option. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

Fig. 4. The new diff view highlights the differences in the dashboard for comparing two files. In this particular case, the two solutions are almost identical, with only minor syntactic differences.

software architecture of the web app (version 2.x). It is intended for researchers, developers, and power users who wish to reuse components in isolation or contribute to the project.

These software components make up the Dolos web app (Fig. 5):

`dolos-core` implements core algorithms of the source code similarity analysis pipeline. An original TypeScript implementation of the winnowing algorithm [15] that is transpiled into a pure ECMAScript Module (ESM) without external dependencies. The ESM package can be executed on any platform that provides a JavaScript runtime engine (web browser, Node.js).

`dolos-parsers` collection of tree-sitter parsers [23] for major programming languages bundled in a single package. Aggregates a collection of Git submodules with a single `node-gyp` configuration to build these parsers and create a single JavaScript package with the resulting node bindings. A custom module aggregating all parsers allows faster integration of additional programming languages into Dolos and

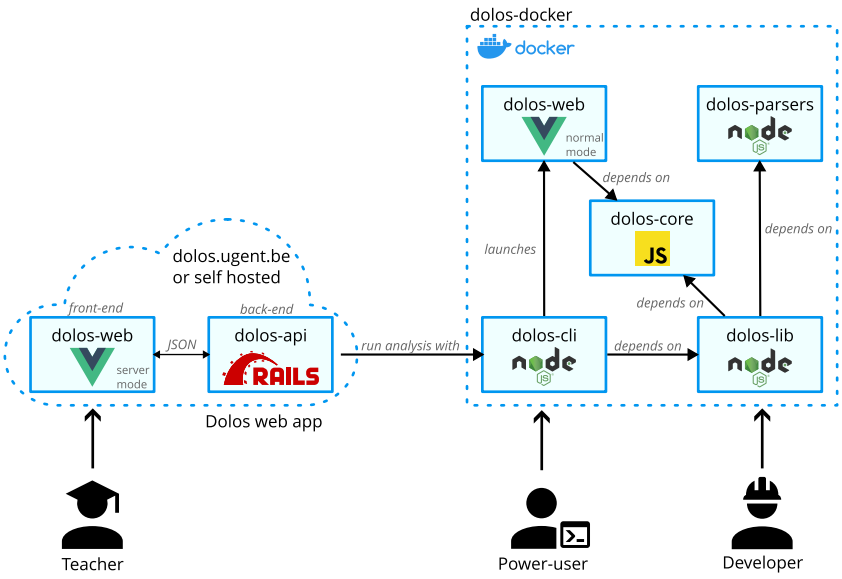

Fig. 5. Diagram of the different components in the Dolos ecosystem and their relationships (Dolos version 2.x). Some components can be used in isolation, as shown by the three users interacting with the components. External dependencies and standalone documentation pages (`dolos-docs`) have been excluded.

keeps supported languages up to date. Avoids dependencies on maintainers of individual parsers to publish new releases on npm.

`dolos-lib` Node.js[7] library for reading source files, parsing source code (depends on `dolos-parsers`), and generating plagiarism reports (depends on `dolos-core`). Supports integration of plagiarism detection into online learning environments. Re-exports algorithms implemented in `dolos-core`.

`dolos-web` web interface implemented on top of the Vue 3[8] JavaScript framework. Provides clean and consistent UX/UI by using Vuetify[9] components. Includes D3-based [24] interactive visualisations from dashboards. Depends on `dolos-core` for client-side execution of some plagiarism analysis pipeline steps (in browser) to keep the app responsive and interactive. May be built in *normal mode* to generate dashboards from an external run of the plagiarism analysis pipeline (used by `dolos-cli`). May be built in *server mode* to add upload functionality onto a *normal mode* build used in the Dolos web app, interacting with the `dolos-api`.

`dolos-cli` Node.js command line interface (CLI) for plagiarism detection functionalities provided by `dolos-lib`. Results from the analysis pipeline can be displayed in the terminal, exported to CSV-files, or launched as dashboards in the browser (depends on `dolos-web`).

`dolos-api` Ruby on Rails[10] web server exposing an application programming interface (API) for plagiarism detection functionalities provided by Dolos. Results are returned in JSON format. For proper sandboxing, each new request uploads source files and runs `dolos-cli` in its own Docker container (`dolos-docker`). The analysis results are stored server-side. The `dolos-web` web interface built in *server mode* communicates with this API to upload datasets, fetch report information, and display the results.

The Dolos documentation website's (https://dolos.ugent.be) source code is included in the `dolos-docs` module. The `dolos-docker` module contains a Docker[11] container pre-installed with the Dolos CLI (`dolos-cli` component). For each new release, a new version of the `dolos-docker` package is automatically published in the GitHub container registry (https://ghcr.io/dodona-edu/dolos). Additionally, new versions of the `dolos-core`, `dolos-parsers`, `dolos-lib`, `dolos-web` and `dolos-cli` packages are automatically published on npm under the `@dodona` scope: `@dodona/dolos-core`, `@dodona/dolos-parsers`, `@dodona/dolos-lib`, `@dodona/dolos-web`, with `@dodona/dolos` providing the `dolos-cli` package.

System requirements

The software components and docker images of Dolos are cross-platform and run on Linux, MacOS and Windows. To run the CLI, Node.js is required with capabilities to install native modules. The CLI runs with 500 MiB of RAM or less and requires more RAM as larger datasets are analysed. We recommend having at least 3 GiB of RAM for hosting a local instance of the web app. The web app can be used in any modern browser (Firefox, Chrome, Edge, ...).

4. Impact

In May 2023, following a "release often/release early" strategy, Ghent University (Belgium) started hosting a first standalone instance of the Dolos web app. At the time of writing (April 2024) this preview version alone has scanned over 5000 collections of source files for possible cases of plagiarism. The significance of source code plagiarism in education is further highlighted by the fact that Dolos has received over 200 stars on GitHub from people from around the world. The code repository also had 52 issues or discussions opened by users outside the core development team to report bugs, ask questions, or suggest features for unsupported use cases.

[7] https://nodejs.org/
[8] https://v3.vuejs.org
[9] https://v3.vuetifyjs.com
[10] https://rubyonrails.org/
[11] https://docker.com

Industry players have begun integrating the Dolos web app into their online learning environments. Codio,[12] an online platform that supports computer science courses, recently switched from using MOSS and JPlag for source code plagiarism detection to a self-hosted instance of Dolos. They justify this decision on their website, stating that: *"Plagiarism detection systems available such as MOSS and JPlag were not developed for university programming courses. Therefore, they can require considerable effort to submit large files of student code projects and to interpret the results. Codio integrates the Dolos plagiarism detection system developed by CS educators for programming courses. This integration provides instructors with enough data and analysis for a lecturer to make a conclusive, final decision. The burden of project data preparation and submission to remote systems such as MOSS and JPlag is removed. The result is a single-click process for the lecturer or teacher"*.

Software components of the Dolos code similarity and clustering pipeline are also being used beyond the original application domain of educational source code plagiarism detection. For instance, in a study on the prevalence of large language models (LLMs) violating software copyright, Yu et al. [25] used Dolos to compare original copyrighted source code with LLM-generated code for. Dolos has also been used for malware detection (personal communication), where k-gram analysis is commonly used to classify computer viruses [26].

5. Conclusions

The latest major release of Dolos (version 2.x) includes a free and open-source web app for educators to detect plagiarism in educational source code. This novel app can be run directly from the browser without any installation or configuration. It is built on top of a state-of-the-art source code similarity detection pipeline that has been optimised for speed and memory consumption. The app supports numerous programming languages out of the box, and the procedure for adding new language parsers has been enhanced. It offers a well-designed set of dashboards for plagiarism analytics. The hierarchical structure of the dashboards enables a thorough examination of suspected plagiarism cases within a collection of source files. Identifying clusters of source files helps comprehend the distribution of plagiarism incidents among groups of students. A comparison of source files side by side can help to identify conclusive evidence that high code similarity is not a coincidence.

Dolos primarily focuses on detecting source code plagiarism in educational settings. However, it has also been utilised for other code similarity and clustering applications, such as malware analysis and generative AI research. We offer comprehensive documentation for power users who wish to host an instance of the web app, integrate plagiarism detection into external learning platforms using its JSON API or JavaScript library, or perform source code similarity analysis from the command line. Dolos' roadmap includes further research into the use of advanced index structures to enable fast scanning of more and longer source files. Additionally, we want to provide specific support for multi-file student projects and take into account the additional longitudinal dimension of students submitting multiple solutions to the same programming exercise. Collaboration on these issues is welcome, and we would be happy to hear about other use cases.

CRediT authorship contribution statement

Rien Maertens: Writing – original draft, Software, Investigation. **Maarten Van Neyghem:** Writing – review & editing, Software, Investigation. **Maxiem Geldhof:** Writing – review & editing, Software, Investigation. **Charlotte Van Petegem:** Writing – review & editing. **Niko Strijbol:** Writing – review & editing. **Peter Dawyndt:** Writing – review & editing, Supervision. **Bart Mesuere:** Writing – review & editing, Supervision, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

No data was used for the research described in the article.

Acknowledgements

Dolos is part of the ecosystem surrounding the Dodona online learning platform[13] [11]. Team Dodona expresses gratitude for the financial support provided by Ghent University (UGent, Belgium) and the Flemish Government (Belgium) through various grants for innovation in education. Additionally, we thank UGent for hosting a free-to-use instance of the Dolos web app (https://dolos.ugent.be). This work was partially supported by the Research Foundation — Flanders (FWO) for ELIXIR Belgium (I002819N, I000323N).

We appreciate all users who reported issues, shared use cases and provided constructive feedback.

Appendix A. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.softx.2024.101755.

References

[1] Sax LJ, Lehman KJ, Zavala C. Examining the enrollment growth: Non-CS majors in CS1 courses. In: Proceedings of the 2017 ACM SIGCSE technical symposium on computer science education. Seattle Washington USA: ACM; 2017, p. 513–8. http://dx.doi.org/10.1145/3017680.3017781.

[2] Balanskat A, Engelhardt K. Computing our future – priorities, school curricula and initiatives across Europe. Tech. rep., European Schoolnet; 2014, p. 28.

[3] UK Department for Education. National curriculum in England: Computing programmes of study. 2013, GOV.UK, URL https://www.gov.uk/government/publications/national-curriculum-in-england-computing-programmes-of-study.

[4] Albluwi I. Plagiarism in programming assessments: A systematic review. ACM Trans Comput Educ 2019;20(1):6:1–28. http://dx.doi.org/10.1145/3371156.

[5] Pierce J, Zilles C. Investigating student plagiarism patterns and correlations to grades. In: Proceedings of the 2017 ACM SIGCSE technical symposium on computer science education. SIGCSE '17, New York, NY, USA: Association for Computing Machinery; 2017, p. 471–6. http://dx.doi.org/10.1145/3017680.3017797.

[6] Cosma G, Joy M. Towards a definition of source-code plagiarism. IEEE Trans Educ 2008;51(2):195–200. http://dx.doi.org/10.1109/TE.2007.906776.

[7] McCabe DL, Trevino LK, Butterfield KD. Academic integrity in honor code and non-honor code environments: A qualitative investigation. J Higher Educ 1999;70(2):211–34. http://dx.doi.org/10.2307/2649128, arXiv:2649128.

[8] Ngo MN. Eliminating plagiarism in programming courses through assessment design. Int J Inform Educ Technol 2016;6(11):873–9. http://dx.doi.org/10.7763/IJIET.2016.V6.808.

[9] Ruiperez-Valiente JA, Alexandron G, Chen Z, Pritchard DE. Using multiple accounts for harvesting solutions in MOOCs. In: Proceedings of the third (2016) ACM conference on learning @ scale. Edinburgh Scotland UK: ACM; 2016, p. 63–70. http://dx.doi.org/10.1145/2876034.2876037.

[10] Novak M, Joy M, Kermek D. Source-code similarity detection and detection tools used in academia: A systematic review. ACM Transactions on Computing Education 2019;19(3):1–37. http://dx.doi.org/10.1145/3313290.

[11] Van Petegem C, Maertens R, Strijbol N, Van Renterghem J, Van der Jeugt F, De Wever B, et al. Dodona: Learn to code with a virtual co-teacher that supports active learning. SoftwareX 2023;24:101578. http://dx.doi.org/10.1016/j.softx.2023.101578.

[12] Weber-Wulff D. Plagiarism detectors are a crutch, and a problem. Nature 2019;567(7749):435. http://dx.doi.org/10.1038/d41586-019-00893-5.

[13] Chuda D, Navrat P, Kovacova B, Humay P. The issue of (software) plagiarism: A student view. IEEE Trans Educ 2012;55(1):22–8. http://dx.doi.org/10.1109/TE.2011.2112768.

[12] https://codio.com

[13] https://dodona.ugent.be

[14] Culwin F, MacLeod A, Lancaster T. Source code plagiarism in UK HE computing schools. In: Proceedings of the 2nd annual LTSN-ICS conference. London, United Kingdom: LTSN Centre for Information and Computer Sciences; 2001, p. 1–7.
[15] Schleimer S, Wilkerson DS, Aiken A. Winnowing: Local algorithms for document fingerprinting. In: Proceedings of the 2003 ACM SIGMOD international conference on management of data. SIGMOD '03, New York, NY, USA: Association for Computing Machinery; 2003, p. 76–85. http://dx.doi.org/10.1145/872757.872770.
[16] Prechelt L, Malpohl G, Philippsen M. Finding plagiarisms among a set of programs with jplag. J UCS 2002;8(11):1016–38. http://dx.doi.org/10.3217/jucs-008-11-1016.
[17] Chandere V, Satish S, Lakshminarayanan R. Online plagiarism detection tools in the digital age: A review. Ann Roman Soc Cell Biol 2021;7110–9.
[18] Jiffriya MAC, Akmal Jahan MAC, Ragel RG. Plagiarism detection tools and techniques: A comprehensive survey. J Sci-FAS-SEUSL 2021;02(02):47–64.
[19] Maertens R, Van Petegem C, Strijbol N, Baeyens T, Jacobs AC, Dawyndt P, et al. Dolos: Language-agnostic plagiarism detection in source code. J Comput Assist Learn 2022;38(4):1046–61. http://dx.doi.org/10.1111/jcal.12662.
[20] Myers EW. An O(ND) difference algorithm and its variations. Algorithmica 1986;1(1):251–66. http://dx.doi.org/10.1007/BF01840446.
[21] Berrezueta-Guzman J, Paulsen M, Krusche S. Plagiarism detection and its effect on the learning outcomes. In: 2023 IEEE 35th international conference on software engineering education and training. 2023, p. 99–108. http://dx.doi.org/10.1109/CSEET58097.2023.00021.
[22] Maertens R, Van Petegem C, Strijbol N, Baeyens T, Jacobs AC, Van Neyghem M, et al. Dolos — Source code plagiarism detection system. 2023, http://dx.doi.org/10.5281/zenodo.7966722, Zenodo.
[23] Brunsfeld M, Hlynskyi A, Qureshi A, Thomson P, Vera J, Turnbull P, et al. Tree-Sitter/tree-Sitter: V0.20.9. 2024, http://dx.doi.org/10.5281/ZENODO.4619183, Zenodo.
[24] Bostock M, Ogievetsky V, Heer J. D^3 data-driven documents. IEEE Trans Vis Comput Graphics 2011;17(12):2301–9. http://dx.doi.org/10.1109/TVCG.2011.185.
[25] Yu Z, Wu Y, Zhang N, Wang C, Vorobeychik Y, Xiao C. CodeIPPrompt: Intellectual property infringement assessment of code language models. In: Krause A, Brunskill E, Cho K, Engelhardt B, Sabato S, Scarlett J, editors. Proceedings of the 40th international conference on machine learning. Proceedings of machine learning research, vol. 202, PMLR; 2023, p. 40373–89, URL https://proceedings.mlr.press/v202/yu23g.html.
[26] Gandotra E, Bansal D, Sofat S. Malware analysis and classification: A survey. J Inform Secur 2014;05(02):56–64. http://dx.doi.org/10.4236/jis.2014.52006.