ProfOlaf: Semi-Automated Tool for Systematic Literature Reviews

Martim Afonso*
INESC-ID and IST, University of Lisbon
Lisbon, Portugal
martim.afonso@tecnico.ulisboa.pt

Nuno Saavedra*
INESC-ID and IST, University of Lisbon
Lisbon, Portugal
nuno.saavedra@tecnico.ulisboa.pt

Bruno Lourenço
INESC-ID and IST, University of Lisbon
Lisbon, Portugal
bruno.horta.lourenco@tecnico.ulisboa.pt

Alexandra Mendes
Faculdade de Engenharia, Universidade do Porto
Porto, Portugal
alexandra@archimendes.com

João F. Ferreira
INESC-ID and Faculty of Engineering, University of Porto
Porto, Portugal
joao@joaoff.com

Abstract

Systematic reviews and mapping studies are critical to synthesize research, identify gaps, and guide future work, but are often labor-intensive and time-consuming. Existing tools provide partial support for specific steps, leaving much of the process manual and error-prone. We present ProfOlaf, a semi-automated tool designed to streamline systematic reviews while maintaining methodological rigor. ProfOlaf supports iterative snowballing for article collection with human-in-the-loop filtering and uses large language models to help select articles, extract key topics, and answer queries about the content of articles. By combining automation with guided manual effort, ProfOlaf enhances the efficiency, quality, and reproducibility of systematic reviews across research fields. ProfOlaf can be used both as a CLI tool and in web application format. A video demonstrating ProfOlaf is available at: https://youtu.be/R-gY4dJlN3s

CCS Concepts

• **Information systems** → **Information retrieval**; **Data management systems**; • **Computing methodologies** → *Information extraction.*

Keywords

Systematic Literature Reviews, Automation, LLM

1 Introduction

Systematic literature reviews and mapping studies play an essential role across research fields, as they organize and synthesize existing knowledge, providing a structured overview that highlights established findings, identifies gaps, and indicates promising directions for future research. Unlike other forms of literature reviews, systematic reviews hold particular scientific value because they follow a transparent, well-defined, and unbiased methodology [4].

Despite their value, conducting systematic reviews is labor-intensive and time-consuming [16, 18]. It requires conducting broad and comprehensive searches in multiple academic databases and careful filtering of large volumes of articles against predefined inclusion and exclusion criteria. Once the collection is complete, reviewers must also analyze the selected studies in depth, identify recurring research topics, and formulate answers to predefined research questions, which demands sustained effort and precision.

Several methods and tools have been proposed to automate or support various steps of the systematic review process [5, 16]. Existing solutions, such as reference managers or search engines, offer only partial assistance, leaving key tasks manual, time-consuming, and error-prone, reducing efficiency and reproducibility [9]. More advanced approaches, such as AI-based tools, have shown promising results [5], but remain fragmented. Although these tools exist for individual steps of the process, no solution provides comprehensive support at all stages. Furthermore, achieving a sensible balance between manual effort and tool assistance is essential to ensure that precision is not compromised in the pursuit of reducing effort.

In response to these requirements, we propose ProfOlaf, a semi-automated tool designed to support and streamline the review process. ProfOlaf implements a structured methodology that adheres to established guidelines for systematic reviews. The methodology was designed with the goal of reducing human effort to the greatest extent possible. For the collection phase, ProfOlaf uses an iterative snowballing process to collect relevant articles, each iteration involving human-led filtering supported by our tool and optionally assisted by large language models (LLMs) as secondary reviewers. In the analysis phase, ProfOlaf integrates LLMs to ease the analysis of the collected articles, enabling users to extract key topics from each article and query the model regarding its contents.

ProfOlaf is open source and available in our repository[1] and in a docker container[2], with our evaluation's artifacts. It can be used both as a command-line interface and a web application.

2 ProfOlaf Overview

The ProfOlaf methodology is illustrated in Figure 1. This section provides a detailed explanation of each step of the pipeline.

Initial Setup. To start a new search with ProfOlaf the user begins by defining a file with an initial seed of article titles, previously selected by the user. The seed can be constructed by following established guidelines in the literature [3, 17]. ProfOlaf receives this and generates a database that stores all the metadata of the articles as well as the intermediate states of each article during the search.

*Both authors contributed equally to this research.

[1] https://github.com/sr-lab/ProfOlaf
[2] https://doi.org/10.5281/zenodo.18386217

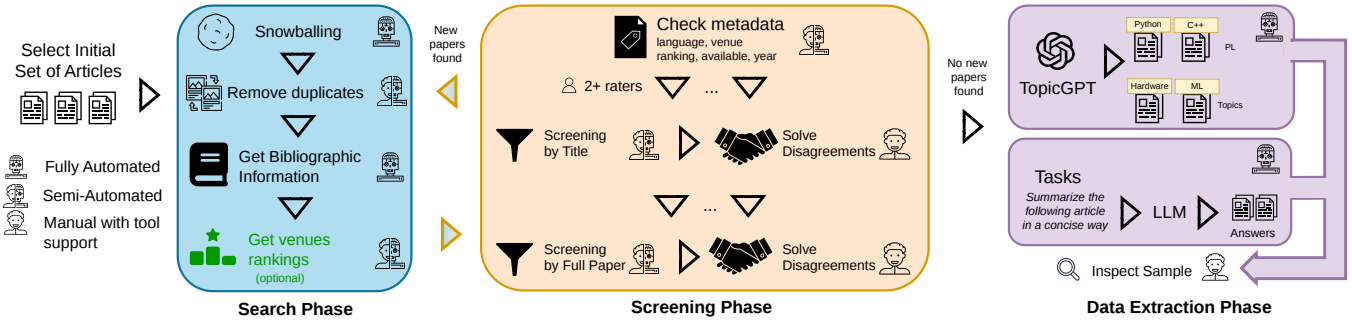

Figure 1: Overview of the ProfOlaf methodology.

Snowballing. We chose snowballing as our primary search method, as prior studies suggest it performs as well as or better than database searches [3, 8]. Keyword-based search is deliberately omitted, allowing us to focus on evaluating the snowballing workflow. After populating the database with the initial set of articles, ProfOlaf retrieves either the citations (*forward snowballing*), the references (*backward snowballing*), or both, for each article. The tool then compiles the bibliographic information for the entire set. ProfOlaf obtains article data from Google Scholar [6], Semantic Scholar [15], and DBLP [14]. However, the tool has been designed to ease the integration of additional search sources. As multiple versions of the same paper may exist under slightly different titles, this stage also includes a mechanism to identify and present these versions, allowing the user to select which one to keep.

Metadata Screening. Subsequently, ProfOlaf filters the retrieved articles using the collected metadata. Filtering criteria include venue ranking, publication year, and language, all of which are optional. Kitchenham et al. recommend such practical criteria to refine article selection [10]. When the venue-ranking filter is applied, the user must execute the step that assigns a ranking to all the venues present in the collected articles. To assist the user, ProfOlaf first uses cosine similarity to find venues previously ranked that are similar to the one being classified. The tool searches for the venue in venue ranking databases and presents the top results for each database. For each result, ProfOlaf outputs the title of the venue, its ranking, and the cosine similarity score of the search. Currently, ProfOlaf searches both in the SCImago and CORE databases, but this functionality is easily extendable to other venue ranking databases.

Article Screening. ProfOlaf adopts the approach outlined by Wohlin, which recommends screening articles progressively, starting with the title, followed by the abstract, and finally the full text [17]. In this phase, the user is prompted to further refine the results, starting by removing irrelevant articles by title and then by their full content. For this, ProfOlaf presents, for each article, its title as well as a *url* where the user can access it. Although the same database instance can be shared among all users, creating a copy for each one allows the process to be carried out in parallel. After each step, the tool can be used to solve discrepancies between the reviewers' assessments. When reviewers conduct their evaluations using separate database instances, these are supplied to the tool and merged for the comparison. The reviewers then discuss the disagreements and reach a consensus, which is recorded by the tool. Until these discussion phases, each reviewer's decisions remain hidden in order to avoid bias. In addition, the tool offers the option of using an LLM as an auxiliary rater to provide additional input during the discussion. Section 3.2 presents an evaluation of this component. With both screening phases completed, the resulting set of articles is used as input to the snowballing phase to start a new iteration. If an iteration does not produce new results, ProfOlaf consolidates the findings from all iterations into a single file.

Topic Modeling and Task Assistant. After collection, ProfOlaf supports article analysis with LLMs. Users can download accessible PDFs (paywalled articles must be retrieved manually), which are then parsed via PyPDF2 and analyzed via topic modeling and a task assistant. All extracted information should be verified by users, either fully or by sampling. The topic modeling module uses TopicGPT [12] to gather different topics from the collection and cluster them according to those topics. TopicGPT is a prompt-based framework that uses LLMs to generate interpretable topics with natural language labels and descriptions. It enables users to group papers for deeper analysis, offering greater transparency and verifiability than traditional bag-of-words approaches. This functionality can also be useful for addressing specific research questions with closed-form answers, such as *"Which programming language is being considered?"*.

The task assistant module enables users to submit an article to a public-access LLM and request tasks such as key information extraction or concise, query-tailored summaries.

3 Evaluation

To evaluate ProfOlaf, we conducted a small illustrative systematic review using Ramos et al. [13] as the seed article. This paper was selected for its Best Paper Award at LLM4Code 2025 and our familiarity with its topic.

3.1 Search and Screening Phase

The screening phase was carried out over seven iterations by two human raters, corresponding to the first and third authors. The following inclusion criteria were applied: (1) related to Machine Learning for Code; (2) written in English; (3) publicly available; and (4) published in a venue ranked by CORE or Scimago.

Table 1: Snowballing process results across iterations.

I	Retrieved	Rejected (Meta. + Screen.)	Approved	Efficiency
1	19	13 + 1	5	0.26
2	100	63 + 7	29	0.29
3	227	158 + 47	22	0.09
4	111	84 + 9	18	0.16
5	100	72 + 3	24	0.24
6	433	414 + 9	10	0.02
7	19	19 + 0	0	0.00
Total	1009	823 + 76	108	0.11

Notes: I = iteration; Meta. = metadata; Screen. = Screening.

Table 2: Automated screening evaluation.

Rater	Screening	Accuracy	Precision	Recall	F1 Score
Human	Title	0.8361	0.8613	**0.9147**	**0.8872**
	Full Content	0.8760	0.9314	**0.9223**	0.9268
LLM	Title	**0.8415**	**0.9098**	0.8605	0.8845
	Full Content	**0.8800**	**0.9417**	0.9151	**0.9282**

The results of all iterations are summarized in Table 1. We calculate the efficiency measure for systematic literature reviews, a metric used by Wohlin [17] to evaluate the amount of noise in the search. Efficiency is calculated as the number of included articles relative to the total number of candidate articles examined. The efficiency by iteration and the final efficiency are both represented in Table 1. After screening, we obtained a total of 108 articles.

3.2 Automated Screening

We used ProfOlaf, configured to use *gpt-5.2*, to perform both screening stages using a concise topic description along with inclusion and exclusion criteria. To evaluate its performance, we compared the model's decisions against the final consensus reached by the two original raters. We also introduced an additional independent rater (fifth author) to screen the articles, allowing us to assess how the LLM's performance compared to that of an individual human. Table 2 presents these findings. The screening process encompassed 183 articles at the title level and 125 articles at the full-text level.

We can observe that the LLM had a performance comparable to that of a human, with slightly greater precision, but lower recall, which indicates that the LLM tended to be more conservative in its screening decisions, favoring correctness over coverage.

3.3 Data Extraction Phase

We selected two tasks for the Topic Modeling module: identifying the programming languages explored and the topics studied; and one task for the Task Assistant module, article summarization. In this evaluation, ProfOlaf was configured to use `gpt-5.2`.

3.3.1 Topics Studied. A set of ground-truth topics was defined by two raters and subsequently revised and assigned during the manual analysis of the article collection. This ground truth is used to evaluate the topic modeling module via two distinct analyses: topic generation and topic assignment.

Topic Generation. Our ground truth consisted of 22 topics. Starting from *Program Repair* as the initial seed, TopicGPT generated 19 topics in its first step, of which 12 matched the ground-truth topics (54%). Notably, two ground-truth topics (*Code Performance* and *Green Software*) were merged into a single topic (*Software Performance and Energy Efficiency*). In addition, one generated topic, *Automated Test Case Generation*, only partially corresponded to our label *Specification Generation* and was therefore excluded from the identified topics. During the refinement step, TopicGPT reduced the set to 14 topics and correctly identified 45% of the ground-truth topics, with the same two mismatches as before. The performance decline is attributable to TopicGPT's removal of infrequent topics; consequently, software tasks that did not frequently appear, such as *Code Translation*, were discarded.

Topic Assignment. Comparing the module's assignments with those of the human raters, we obtained a precision of 0.645 and a recall of 0.850.

An analysis of the resulting metrics reveals that the *Benchmarks* label accounted for the highest number of incorrect assignments, with 30 false positives. 47.6% of the predictions assigned to this label were incorrect. This is explained by the frequent mention of benchmark suites in many papers, even when they do not represent a core contribution, which leads the model to over-assign this topic. Addressing these alone would result in an estimated precision improvement of approximately 9%.

In contrast, while the *Human–AI Collaboration* label produced fewer false positives in absolute terms, it exhibited the highest misidentification rate (76.9% of false positives). This trend is attributed to the model's tendency to assign this label to articles that merely reference human comparisons or interactions, rather than focusing on substantive contributions or studies centered on Human–AI collaboration in software engineering.

3.3.2 Programming Languages Used. As before, during the analysis of the papers, the programming languages were manually identified. For this task, the topic modeling module identified 40 programming languages, while the raters identified 47. The module achieved an average precision of 0.590 and an average recall of 0.710.

Manual inspection of the assignments revealed that the model frequently tagged Python as one of the languages. This likely stems from the fact that many papers employ Python for data processing or model training, leading to its assignment even when it was not the language under exploration. Additionally, the model often assigned more languages than necessary, suggesting a tendency toward over-assignment and a degree of hallucination.

3.3.3 Task Assistant. For the Task Assistant task, we asked the model for a summary of each article. Summaries were evaluated by two raters according to four parameters on a Likert scale from 1 to 5. The results of the evaluation are presented in Table 3.

Both raters generally agreed the summaries were accurate and free from hallucinations. *Coverage* was the lowest-scoring criterion, as some summaries did not fully capture the most important details; however, this limitation was minor (average score of 4.333). The remaining criteria received high scores, suggesting that summaries were typically well-structured, easy to follow, and conveyed key points with minimal redundancy. Coherence/Structure had the highest standard deviation, indicating that the model's ability to organize ideas varied the most across summaries.

Table 3: Summary Quality Evaluation.

Criterion	Mean	Std	Criterion	Mean	Std
Faithfulness	4.907	0.242	Structure	4.558	0.454
Salience	4.333	0.367	Conciseness	4.648	0.334

3.3.4 Discussion. The results suggest that while human raters are still essential during the screening process, employing an LLM as an additional rater can provide information, stimulate discussion, and support the refinement of inclusion and exclusion criteria during the snowballing phase.

For complex tasks such as topic modeling, we infer that current LLMs and associated methodologies are not yet reliable enough to operate autonomously. Their effectiveness is maximized in a human-in-the-loop setting, serving as assistive tools rather than independent decision-makers. LLMs can generate an initial set of topics and corresponding assignments, which are then validated and refined by a human reviewer. This substantially reduces manual effort while maintaining methodological reliability. Notably, the models did not hallucinate and generated plausible topics, but these often diverged from the task definition. More task-aligned prompt design to improve topic quality and relevance is left as future work.

For less demanding tasks such as summarization, LLMs already demonstrate satisfactory performance, making them well suited for direct use with minimal human intervention, saving time during the article analysis process.

4 Related work

Bacinger et al. present a semi-automated system for literature review search and screening, automating article retrieval, search-term definition, and relevance ranking with machine learning, and supporting curation and export of the final set of papers [2]. Unlike their tool, which relies on database searches, ProfOlaf uses snowballing, a method shown to perform as well as or better than database searches [3, 8], and supports data extraction from articles.

Agarwal et al. introduce LitLLM, an LLM-based toolkit for the generation of scientific literature reviews. It automatically generates search keywords from user-provided abstracts, retrieves and re-ranks relevant papers, and produces related work text grounded in these papers through Retrieval-Augmented Generation [1]. Despite its text generation power, LitLLM lacks manual curation or snowballing, unlike ProfOlaf's human-in-the-loop workflow.

He et al. propose PaSa, an LLM-based agent for academic paper search. PaSa automates query generation, retrieves and expands results through citation networks, and uses a selector agent to evaluate relevance, thereby enabling comprehensive and accurate literature retrieval [7]. The authors' work emphasizes automated retrieval and screening but does not allow manual curation, whereas ProfOlaf balances automation with the researcher's control.

Li et al. present ChatCite, a tool that automates literature summarization by extracting key elements from papers, generating comparative summaries through an iterative reflective process, and evaluating the results with a novel automatic metric called G-score [11]. Unlike ProfOlaf, which covers the review cycle from the search to the data extraction phase, ChatCite focuses exclusively on summary generation and is therefore complementary.

Several tools support the systematic literature review process. Covidence[3] provides an end-to-end platform for managing reviews, including reference import, screening, and data extraction. Rayyan[4] focuses on collaborative title and abstract screening with filtering and machine-learning–assisted suggestions, while ASReview[5] applies active learning to prioritize relevant studies during screening. In contrast, our tool supports a broader portion of the literature review workflow, including automated snowballing, metadata filtering, collaborative screening with disagreement resolution, and LLM-assisted analysis. Additionally, ProfOlaf is free and open source, facilitating transparent and reproducible review workflows.

5 Limitations

The evaluation was conducted in a single domain using one seed article due to time constraints, limiting the generalizability of the results. Future evaluations should consider multiple domains and initial seeds to better assess the robustness of the approach.

The current workflow relies on snowballing, which proved effective in our experiments but restricts users to a single search strategy. Future versions will support additional search methods to provide greater flexibility.

ProfOlaf also depends on third-party APIs for data retrieval, meaning changes in their availability, usage limits, or policies may affect the tool's functionality.

Finally, the topic modeling module already offers useful exploratory insights, supporting users in navigating the literature. Although further improvements are needed for fully rigorous analytical use in systematic literature reviews, the module already provides a strong foundation for future refinement and evaluation.

6 Conclusion

ProfOlaf addresses key challenges in the systematic review process, combining iterative snowballing with LLM-assisted screening and analysis, striking a balance between automation and human oversight. By improving efficiency, rigor, and reproducibility, it enables researchers to conduct higher-quality reviews with reduced effort. Given the rapidly growing volume of research in software engineering, ProfOlaf can be valuable in helping SE researchers keep pace with the field.

Acknowledgments

This work was supported by national funds through Fundação para a Ciência e a Tecnologia, I.P. (FCT) under grants 2023.18565.PRT and BD/04736/2023, and projects UID/50021/2025 (DOI: 10.54499/UID/50021/2025), UID/PRR/50021/2025 (DOI: UID/PRR/ 50021/2025). This work was also co-funded by the European Union through the Lisboa 2030 Programme (ERDF) and by national funds through FCT, I.P., under project no. 15018 (SafeIaC project, DOI: 10.54499/2023.18089.ICDT).

References

[1] Shubham Agarwal, Gaurav Sahu, Abhay Puri, Issam H Laradji, Krishnamurthy DJ Dvijotham, Jason Stanley, Laurent Charlin, and Christopher Pal. 2024. Litllm: A toolkit for scientific literature review. *arXiv preprint arXiv:2402.01788* (2024).

[3] https://www.covidence.org/
[4] https://www.rayyan.ai/
[5] https://asreview.nl/

[2] Filip Bacinger, Ivica Boticki, and Danijel Mlinaric. 2022. System for semi-automated literature review based on machine learning. *Electronics* 11, 24 (2022), 4124.

[3] Deepika Badampudi, Claes Wohlin, and Kai Petersen. 2015. Experiences from using snowballing and database searches in systematic literature studies. In *Proceedings of the 19th international conference on evaluation and assessment in software engineering*. 1–10.

[4] Pearl Brereton, Barbara A Kitchenham, David Budgen, Mark Turner, and Mohamed Khalil. 2007. Lessons from applying the systematic literature review process within the software engineering domain. *Journal of systems and software* 80, 4 (2007), 571–583.

[5] Nicholas Fabiano, Arnav Gupta, Nishaant Bhambra, Brandon Luu, Stanley Wong, Muhammad Maaz, Jess G Fiedorowicz, Andrew L Smith, and Marco Solmi. 2024. How to optimize the systematic review process using AI tools. *JCPP advances* 4, 2 (2024), e12234.

[6] Google. 2025. Google Scholar. https://scholar.google.com Accessed 26-09-2025.

[7] Yichen He, Guanhua Huang, Peiyuan Feng, Yuan Lin, Yuchen Zhang, Hang Li, et al. 2025. Pasa: An llm agent for comprehensive academic paper search. *arXiv preprint arXiv:2501.10120* (2025).

[8] Samireh Jalali and Claes Wohlin. 2012. Systematic literature studies: database searches vs. backward snowballing. In *Proceedings of the ACM-IEEE international symposium on Empirical software engineering and measurement*. 29–38.

[9] Staffs Keele et al. 2007. *Guidelines for performing systematic literature reviews in software engineering*. Technical Report. Technical report, ver. 2.3 ebse technical report. ebse.

[10] Barbara Kitchenham, Stuart Charters, et al. 2007. Guidelines for performing systematic literature reviews in software engineering. (2007).

[11] Yutong Li, Lu Chen, Aiwei Liu, Kai Yu, and Lijie Wen. 2025. ChatCite: LLM Agent with Human Workflow Guidance for Comparative Literature Summary. In *Proceedings of the 31st International Conference on Computational Linguistics*. 3613–3630.

[12] Chau Minh Pham, Alexander Hoyle, Simeng Sun, Philip Resnik, and Mohit Iyyer. 2024. Topicgpt: A prompt-based topic modeling framework. DOI: 10.48550. *arXiv preprint arXiv.2311.01449* (2024).

[13] Daniel Ramos, Claudia Mamede, Kush Jain, Paulo Canelas, Catarina Gamboa, and Claire Le Goues. [n. d.]. Are large language models memorizing bug benchmarks?, 2024. *URL https://arxiv. org/abs/2411.13323* 6 ([n. d.]).

[14] Schloss Dagstuhl - Leibniz Center for Informatics. 2025. DBLP. https://dblp.org Accessed 26-09-2025.

[15] Semantic Scholar. 2025. Semantic Scholar. https://www.semanticscholar.org Accessed 26-09-2025.

[16] Raymon Van Dinter, Bedir Tekinerdogan, and Cagatay Catal. 2021. Automation of systematic literature reviews: A systematic literature review. *Information and software technology* 136 (2021), 106589.

[17] Claes Wohlin. 2014. Guidelines for snowballing in systematic literature studies and a replication in software engineering. In *Proceedings of the 18th international conference on evaluation and assessment in software engineering*. 1–10.

[18] He Zhang and Muhammad Ali Babar. 2013. Systematic reviews in software engineering: An empirical investigation. *Information and software technology* 55, 7 (2013), 1341–1354.