HALLUCITECHECKER: A Lightweight Toolkit for Hallucinated Citation Detection and Verification in the Era of AI Scientists

Yusuke Sakai, Hidetaka Kamigaito, Taro Watanabe
Nara Institute of Science and Technology (NAIST), Japan
{sakai.yusuke.sr9, kamigaito.h, taro}@is.naist.jp

Abstract

We introduce HALLUCITECHECKER, a toolkit for detecting and verifying hallucinated citations in scientific papers. While AI assistant technologies have transformed the academic writing process, including citation recommendation, they have also led to the emergence of hallucinated citations that do not correspond to any existing work. Such citations not only undermine the credibility of scientific papers but also impose an additional burden on reviewers and authors, who must manually verify their validity during the review process. In this study, we formalize hallucinated citation detection as an NLP task and provide a corresponding toolkit as a practical foundation for addressing this problem. Our package is lightweight and can perform verification in seconds on a standard laptop. It can also be executed entirely offline and runs efficiently using only CPUs. We hope that HALLUCITECHECKER will help reduce reviewer workload and support organizers by enabling systematic pre-review and publication checks. Our code is released under the Apache 2.0 license on GitHub[1] and is distributed as an installable package via PyPI[2]. A demonstration video is available on YouTube[3].

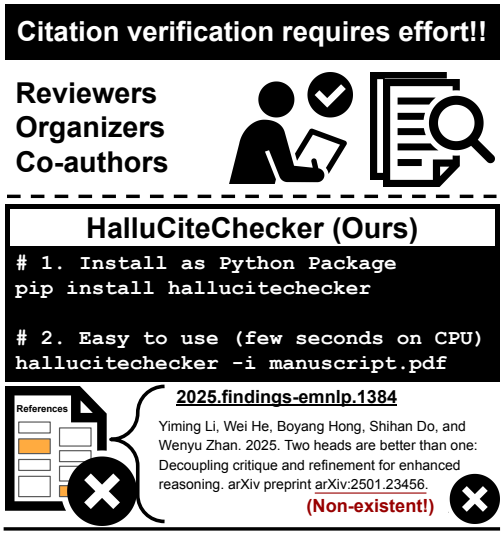


Figure 1: Illustration of the usage scenario of HALLUCITECHECKER. Hallucinated citations are often disguised among many correct citations, making manual verification labor-intensive. HALLUCITECHECKER is easy to install and works out-of-the-box on a laptop. It highlights candidate hallucinated citations, enabling efficient verification, promoting author awareness, and reducing reviewer workload. In the example paper (Ji and Lu, 2025), more than ten citations were identified.

1 Introduction

Citations are a fundamental component of scientific papers (Wang and Barabási, 2021). They serve to substantiate claims with evidence, acknowledge prior work, and clarify the positioning and novelty of new contributions within a research area. Beyond supporting scientific arguments, citations also function as a key quantitative metric for evaluating scientific impact. Citation counts are widely used to assess the influence of papers, researchers, and publication venues, forming the basis of metrics such as the h-index and impact factor. As a result, citations serve not only as scholarly references but also as a mechanism for allocating academic credit.

Recently, reflecting the advancement of AI for Science and AI Scientist technologies, writing assistance powered by large language models (LLMs) has transformed the academic writing process, including automated drafting, paraphrasing, and citation recommendations. These tools improve productivity and accessibility, particularly for non-native English speakers, junior researchers, and students (Kusumegi et al., 2025). However, they have also introduced new risks. Hallucinated citations, which are references that do not correspond to any real work or contain incorrect bibliographic information, have begun to appear increasingly in papers under review, preprints, and even peer-

[1] : yusuke1997/HalluCiteChecker
[2] : pip install hallucitechecker
[3] : https://youtu.be/WG3WXgH2pok

Proceedings of the 15th International Joint Conference on Natural Language Processing and the 5th Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics, pages 33–41
November 6-10, 2026 ©2026 Association for Computational Linguistics

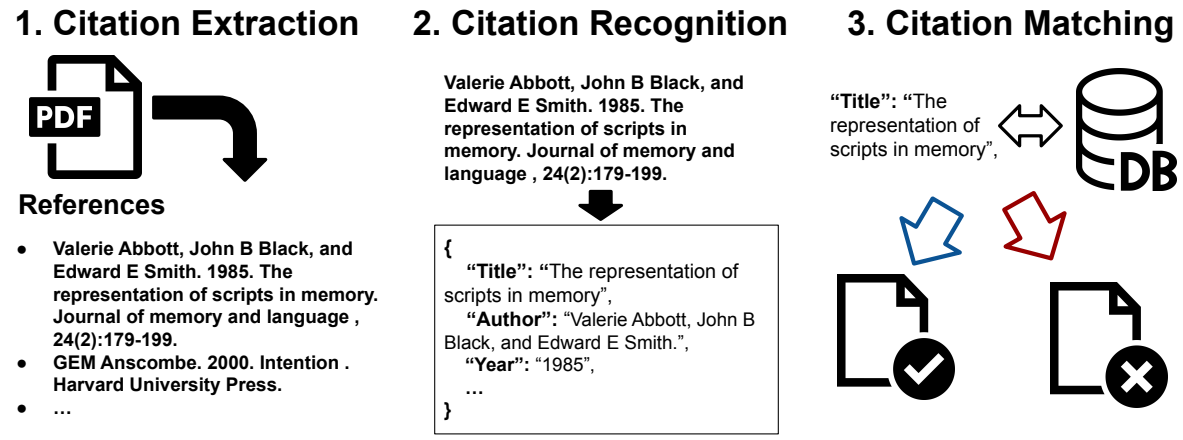


Figure 2: Overview of the hallucinated citation detection task, which consists of three subtasks: Citation Extraction (Section 2.1), Citation Recognition (Section 2.2), and Citation Matching (Section 2.3).

reviewed publications. Recent reports indicate that their prevalence has increased rapidly (Sakai et al., 2026; Naddaf and Quill, 2026). This trend poses a serious threat to the integrity and reliability of scientific communication, which has traditionally relied on an implicit assumption of good faith, where authors accurately cite existing and relevant work[4].

The growing presence of hallucinated citations has shifted additional verification responsibilities onto reviewers, who must now manually confirm the validity of cited references to maintain publication quality. This process imposes a significant and unilateral burden on reviewers and publication organizers. Consequently, papers containing hallucinated citations may face desk rejection or other corrective actions (Resnik and Hosseini, 2026; Liakata et al., 2026). Despite these efforts, systematic investigations have revealed that papers containing hallucinated citations have been accepted to several venues, highlighting the limitations of manual verification and the need for automated detection.

To address this, we formalize hallucinated citation detection as a novel NLP task and introduce HALLUCITECHECKER, a lightweight toolkit for verifying and detecting hallucinated citations in scientific papers, as shown in Figure 1. Our system is designed to be practical and accessible. It can verify within seconds on a standard laptop and runs entirely offline, using only CPUs. This enables seamless integration into existing workflows, e.g., pre-submission checks by authors, reviewer assistance during peer review, and automated screening by organizers and editors. It also adopts a modular design that enables easy implementation and extension within a unified framework. By providing an efficient and deployable solution for hallucinated citation detection, HALLUCITECHECKER aims to reduce reviewer workload, improve publication reliability, and help preserve the integrity of scientific communication in the era of AI-assisted research.

2 Task Definition

We define the Hallucinated Citation Detection task as a structured task consisting of three subtasks: Citation Extraction, Citation Recognition, and Citation Matching. Figure 2 shows the overview of these subtasks. This decomposition clarifies the challenges involved in detecting hallucinated citations and highlights opportunities for systematic improvement and future development. Moreover, by organizing previously scattered approaches into a unified task formulation, our framework facilitates systematic comparison, implementation, and future development.

2.1 Citation Extraction

Scientific papers are typically distributed as structured documents, e.g., PDF files[5]. Under this con-

[4]Erroneous citations have long been known to exist in scientific literature (Wang and Barabási, 2021; Sweetland, 1989; Xu et al., 2026). While such errors were traditionally attributed to human mistakes, the LLM era has seen a rapid increase in hallucinated citations, including fabrication of non-existent works (Xu et al., 2026; Oladokun et al., 2025; Walters and Wilder, 2023). In addition, minor inconsistencies, e.g., slight variations in bibliographic details, have also become more common (Bianchi et al., 2025; Bienz et al., 2026). Regardless of their origin, it is desirable to systematically detect them.

[5]Scientific research is also presented in various formats, including physical printouts and HTML pages such as blog

straint, the first step is to extract citation information from the document. Citation extraction involves identifying and parsing bibliographic entries from raw documents, which presents challenges due to variations in layout, formatting, and document quality. This subtask offers several directions for improvement and extension. Recognition accuracy and efficiency can be improved through parsing, OCR, and post-correction. Moreover, it is necessary to isolate citation-specific information, such as identifying reference sections and separating individual bibliographic records.

In addition, this subtask can be extended to extract peripheral information, including structural information, e.g., bounding-box coordinates, and contextual information associated with citations. Such extensions enable a structural understanding and visualization of citation usage, and provide additional signals for downstream recognition and verification, including assessing whether citations are appropriate and valid. In summary, this subtask focuses on extracting citation-related information, including bibliographic entries and associated structural and contextual metadata, from documents.

2.2 Citation Recognition

Citation information is typically represented as inline text within documents. Therefore, it is necessary to identify and classify individual citation components, such as the title, author names, publication year, venue, and page numbers. Citation styles vary widely, including standard formats such as APA and MLA, as well as conference- and journal-specific formatting conventions. In addition, citation entries may contain missing or incomplete fields. Consequently, naive rule-based extraction approaches are insufficient in real-world scenarios.

This subtask aims to identify the semantic structure of citation text in the wild by determining which spans correspond to specific bibliographic components. This can be regarded as a specialized form of named entity recognition (NER) task, where the goal is to extract structured bibliographic fields from raw citation strings. Although existing NER techniques can be adapted for scientific documents, challenges remain due to the large number of entity types and the diversity of citation formats encountered across venues and disciplines.

2.3 Citation Matching

Given structured citation information, we determine whether each citation is valid by matching it against bibliographic databases, such as local databases and web-based scholarly search engines. While it is possible to use various citation fields for this purpose, prior studies on hallucinated citation verification have shown that title information serves as the reliable matching key (Sakai et al., 2026; Bienz et al., 2026; Xu et al., 2026)[6]. Therefore, in this paper, we primarily focus on title-based citation matching[7]. Note that if structured bibliographic data, such as BibTeX entries, are available, citation verification can be conducted directly starting from this subtask. However, such structured metadata is rarely available in practice. Therefore, it is desirable to consider the entire pipeline, including citation extraction, recognition, and matching.

3 HALLUCITECHECKER: Principles

We design our toolkit, HALLUCITECHECKER, with the following five design principles in mind:

Easy to Use: The toolkit should be easy to install and use, avoiding complex setup procedures. To achieve this, we provide the toolkit as a Python package that can be installed with a single command and executed through a one-line command. Our goal is to enable out-of-the-box functionality without requiring additional configuration, similar to widely used tools such as ACL PUBCHECK[8].

Lightweight: The toolkit should be practical and efficient in typical usage environments. Therefore, it is designed to run efficiently even on standard laptop resources within practical time constraints.

Offline: For instance, during the peer-review process, uploading manuscripts to external APIs or

posts. In this paper, we focus on PDF documents for simplicity of exposition. These alternative formats are merely variations in input format. Moreover, reliably extracting complete and accurate citation information remains challenging, despite significant progress in document parsing research.

[6] One possible reason is that author names often appear in abbreviated or truncated forms, such as the use of initials, omission of middle names, or the use of "et al." to represent multiple authors. In addition, venue names are frequently abbreviated, and bibliographic fields, e.g., page numbers, may be incomplete or inconsistently formatted. While it is possible to require identifiers such as DOIs or URLs, these identifiers are not consistently available for all citations. Moreover, DOI resolution pages vary in structure and formatting, making it difficult to retrieve reliable metadata from them. In contrast, title information is typically more distinctive, stable, and presented in its full form, making it a practical and reliable key.

[7] This subtask is not limited to this setting. For example, matching strategies and classification criteria can be explored. We expect this to be one of the most active research directions.

[8] https://github.com/acl-org/aclpubcheck

web services may introduce risks of data leakage and violate confidentiality policies. To address these concerns, the toolkit is designed to function fully offline without relying on external services. While optional external integrations may be supported, the core functionality remains fully operational in isolated, in-house environments.

Non-Generative AI: Generative AI tools, such as LLMs, may be restricted or prohibited under certain peer-review policies[9]. Reliance on such tools could introduce compliance, reproducibility, and hallucination concerns. Hence, the core implementation avoids the use of generative AI and instead relies on standard NLP and ML techniques.

Simple Implementation: It follows a minimalist design to facilitate maintainability and extensibility. By providing well-decomposed components and a clear implementation structure with explicit separation between base classes and specific implementations, and enabling easy following of the data and processing flow, the toolkit supports easy integration of new methods with minimal effort.

4 HALLUCITECHECKER: System Details

We construct a framework where the three subtasks defined in Section 2 are implemented as individual components, as shown in Listing 1. Each class corresponds to a key subtask. This modular design supports easy extension and enables flexible integration of new methods for future research. The pipeline is controlled through a unified `Citation` dataclass, which manages data flow across components and provides a clear and structured workflow. Based on this design, we primarily describe the default out-of-the-box implementations.

4.1 Citation Extractor

Given a PDF document as input, this component extracts the raw text of each citation and returns a list of `Citation` objects. We use `Docling` (Auer et al., 2024) to perform PDF parsing using only CPU resources[10]. Next, we heuristically identify and extract the reference section, and then apply post-processing to reconstruct citation strings that may be split across page breaks or line boundaries. This process yields citation information as one-line raw text strings. In addition, structural metadata such as bounding box coordinates is obtained. The `Docling` is available under the MIT License.

```python
from abc import ABC, abstractmethod
from typing import List
from dataclasses import dataclass, field

'### Section 4.4: Dataclass `Citation` ###'
@dataclass(slots=True)
class Citation:
  raw_text: str
  bboxes: list = field(default_factory=list)
  title: str = ""
  author: str = ""
  ...

'### Section 4.1: Citation Extractor ###'
class BaseCitationExtractor(ABC):
  @abstractmethod
  def extract(path: str) -> List[Citation]:
    # Extract each citation from the PDF and return it as a List[Citation].
    pass
    # [Citation(raw_text, bboxes, ...), ...]
  ...

'### Section 4.2: Citation Recognizer ###'
class BaseCitationRecognizer(ABC):
  @abstractmethod
  def parse(citation: Citation) -> Citation:
    # Parse the citation string and store the bibliographic information.
    pass
    # Citation(raw_text='', bboxes= [...], title='', author='', ...)

  def batch_parse(citations: List[Citation], ...) -> List[Citation]:
    # In practice, more batchfied them.
    return [self.parse(c) for c in citations]

'### Section 4.3: Citation Matcher ###'
class BaseCitationMatcher(ABC):
  @abstractmethod
  def is_match(self, c: Citation) -> bool:
    # Matching via databases or APIs.
    # If necessary, store the DBs in __init__
    pass

  def verify(citations: List[Citation]) -> List[Citation]:
    # If you want to batch process or return rich information, override `verify`.
    return [c for c in citations if not self.is_match(c)]
```

Listing 1: Pseudo-code of HALLUCITECHECKER to illustrate our design. It is simplified for this paper based on the base-class components. Each class corresponds to a subtask in our pipeline. Therefore, the system works as long as the derived codes comply with the input and output interfaces.

[9] E.g., the ICML 2026: https://icml.cc/Conferences/2026/LLM-Policy, and the ACL Rolling Review Policy: https://aclrollingreview.org/reviewerguidelines#q-can-i-use-generative-ai [Accessed on Jul. 15th, '26]

[10] `Docling` provides optional OCR and LLM-based correction features that may utilize GPU resources. However, we use only its internal CPU-based parsing module, *docling-parse*, as a front-end tool for PDF parsing.

4.2 Citation Recognizer

Given citation strings obtained from Section 4.1, this component extracts the title information. However, heuristic approaches based on regular expressions have inherent limitations due to variations in citation formats (Bienz et al., 2026). Therefore, citation recognition is formulated as a sequence labeling task similar to NER, where each token in the citation string is tagged to identify structured bibliographic fields, including the title.

We employ DeLFT (DeL, 2018–2024) to extract structured bibliographic fields, including the title, from citation strings[11]. Specifically, we use a pretrained `BidLSTM_CRF_FEATURES` model (Lample et al., 2016; Chiu and Nichols, 2016) with `glove840B` embeddings (Pennington et al., 2014). Given a citation string, the model performs sequential labeling with 18 entity tags[12], from which the title field is extracted. The tools used in this component are available under the Apache 2.0 License.

4.3 Citation Matcher

Given the extracted title information, this component verifies whether the cited paper exists by matching it against bibliographic databases. Based on Sakai et al. (2026); Xu et al. (2026), we employ character-level fuzzy matching on citation titles using similarity scores based on the normalized Levenshtein distance implemented in the `RapidFuzz` library (Bachmann, 2025), with a threshold of 0.9. We also include bibliographic databases derived from ACL Anthology, arXiv, and DBLP as part of the default setting, following Sakai et al. (2026).

Although Listing 1 shows a simplified implementation that returns a boolean verification result, the system can be easily extended to return similarity scores or additional metadata by overriding the `verify()` function. While external APIs can be useful for this task, to comply with our design principles stated in Section 3, we do not include external API access in the core module. Instead, external API access is treated as optional, and citation matching is primarily performed against internal bibliographic databases. The libraries used in this component are available under the MIT License.

4.4 Unified Dataclass: `Citation`

As shown in Listing 1, each component works by adding information to a unified `Citation` dataclass, which serves as the central data structure for managing all citation-related information. In particular, representing each citation as a single `Citation` object from the beginning eliminates the need to assign or manage sequential identifiers, resulting in a simpler and more maintainable implementation. This design also enables peripheral information obtained in earlier stages, such as bounding box coordinates, to be preserved and accessed by downstream components without requiring intermediate steps to manage such metadata.

Furthermore, processing decisions are made based on the presence of required fields, and undefined attributes are treated as errors. This design enforces a predefined and type-safe data structure, enabling robust handling of incomplete or exceptional cases without disrupting the overall workflow. Moreover, the dataclass provides utility methods such as `from_dict` and `display`, which encapsulate data formatting and representation within the dataclass itself. This explicit separation between data representation and pipeline processing facilitates tracing and debugging across pipeline stages while minimizing implementation complexity.

4.5 Logistics and Infrastructures

Pretrained weights and databases. While the HALLUCITECHECKER codebase can be easily installed via PyPI, certain resources used in Sections 4.2 and 4.3 are too large to be distributed directly within the package. Although users can initialize and update these resources locally, we provide out-of-the-box access to pretrained weights and bibliographic databases to ensure ease of use and efficient loading. To minimize loading latency, the `glove840B` embeddings are distributed through the Hugging Face Hub (Wolf et al., 2020) in Lightning Memory-Mapped Database (LMDB) format, enabling efficient memory-mapped access. The citation databases are distributed using Hugging Face Datasets (Lhoest et al., 2021), which leverage Apache Arrow for fast loading and efficient storage. These databases are automatically versioned and updated without requiring manual intervention.

[11] Prior analysis work (Sakai et al., 2026; Xu et al., 2026) has used GROBID (GRO, 2008–2026), a de facto tool for scholarly document parsing. However, GROBID is implemented in Java and typically requires setting up a Docker environment, which does not align with our Easy-to-Use design principle, described in Section 3. Therefore, to meet our requirements, we use DeLFT, a Python-based reimplementation of GROBID's sequence-labeling models.

[12] 18 entities are as follows: author, booktitle, collaboration, date, editor, institution, issue, journal, location, note, pages, publisher, pubnum, series, tech, title, volume, and web.

```
hallucitechecker -i ok.pdf hallucinated.pdf \
                 [-o outputs] ...
# All Clear!  # -> ok.pdf
# {'author': 'Yiming Li, Wei He, ...,
#   'title': 'Two heads are better than...'
# }, ...      # -> hallucinated.pdf
```

Listing 2: CLI-based usage of HALLUCITECHECKER.

Developer tools. We provide developer tools to facilitate performance measurement and system evaluation. This includes a stopwatch utility for measuring processing time (Deguchi et al., 2024), testing utilities with CI/CD integration to support systematic comparison and validation of improvements, and profiling tools that export execution environment information such as memory usage.

PDF highlighting for review support. We support the processing of multiple PDF documents and provide functionality to generate highlighted output PDFs indicating detected hallucinated citation candidates. To support fair and consistent review and publication checks, the toolkit allows fixed database versions to be used and shared among reviewers, organizers, and authors. This ensures reproducible verification results and facilitates efficient consensus-building for handling hallucinated citations, including desk rejection decisions. If false positives occur, authors can provide supporting documentation to justify the citation, thereby preventing inappropriate desk rejection. This enables more efficient and transparent verification while maintaining fairness in the review process.

5 Try!: `pip install hallucitechecker`

Case 1: CLI-based verification. Listing 2 shows example usage via the command-line interface (CLI), which enables verification of multiple PDF files as input. The toolkit displays *All Clear!* if no hallucinated citations are detected. Otherwise, detected candidates are reported in the CLI output. The toolkit also provides optional features. For example, by setting an output directory, highlighted PDFs can be generated with detected citation candidates visually marked, as illustrated in Figure 1.

Case 2: Python API-based customization. As shown in Listing 3, each component can be used independently via the Python API, enabling flexible customization. The toolkit follows a one-function, one-object design principle, allowing components to be easily composed in multi-stage pipelines. For example, users can apply multiple databases sequentially by chaining instances of the component.

```
from HalluCiteChecker import *

doc_extractor = DoclingCitationExtractor()
doc_extractor.initialize()
citation_parser = DelftCitationRecognizer()

# Use your own database: (csv, tsv, Datasets)
# Require fields: title [, id]
db1 = FuzzCitationMatcher(db_path="...",
  score_cutoff=90, ...)
db2 = FuzzCitationMatcher(db_path="...", ...)
api_db = YourCustomAPICitationMatcher(...)

path ='your_manuscripts.pdf'
# Section 4.1: Citation Extractor
refs = doc_extractor.extract_references(path)
# Section 4.2: Citation Recognizer
citations = citation_parser.parse_batch(refs)
# len(citations) -> 50

# Section 4.3: Citation Matcher (customized)
citations = db1.verify(citations)
citations = db2.verify(citations)
citations = api_db.verify(citations)
# len(citations) -> 1 (hallucinated citation)
```

Listing 3: An example of the current implementation demonstrating the use of custom databases or APIs.

	Main (Long)	Main (short)	Findings	Total
#Paper	16	1	10	27

Table 1: Number of papers containing hallucinated citations newly identified by HALLUCITECHECKER. These citations were missed by the commercial screening tool WisPaper, indicating that HALLUCITECHECKER can detect such cases.

6 Evaluation

Performance. First, we confirmed that the toolkit successfully detects hallucinated citations in the EMNLP 2025 Main track reported in Sakai et al. (2026). This indicates that it achieves at least comparable recall to that of prior analyses. Next, we applied it to the ACL 2026 Main and Findings papers, which had already been screened by the organizers using WisPaper before publication (Liakata et al., 2026). Table 1 shows the additional hallucinated citations identified by our toolkit, and Table 2 shows examples of the actual detected examples, suggesting higher recall than the commercial system. Furthermore, Badalova and Mayr (2026) compare similar hallucinated citation detection tools and report detailed performance metrics, including precision. Due to space limitations, we refer

Paper ID	Evidence
2026.acl-long.1393 Xue et al. (2026)	Edoardo Maria Ponti, Clara Vania, Goran Glavaš, Olga Majewska, Zining Wu, Jannis Lin, Ivan Vulic, and Anna Korhonen. 2023. Fine-tuning language models for specific tasks can be harmful. arXiv preprint arXiv:2310.09419.
2026.findings-acl.828 Wang et al. (2026)	Yaowei Zheng, Junting Lu, Shenzhi Wang, Zhangchi Feng, Dongdong Kuang, and Yuwen Xiong. 2025. Easyr1: An efficient, scalable, multi-modality rl training framework. arXiv preprint arXiv:2501.12345.

Table 2: Representative examples of hallucinated citations detected by HalluCiteChecker. Some hallucinated citations match unrelated papers when searched by their arXiv IDs. The full list is available in the spreadsheet.

Time	Extractor	Recognizer	Matcher	Total
MacBook Pro *(Apple M3 chip, 24GB RAM, macOS 15.7.3)*				
msec/paper	6206.8ms	215.9ms	798.1ms	7220.9ms
msec/citation	121.7ms	4.2ms	15.6ms	141.6ms
Total (sec)	620.7s	21.6s	79.8s	722.1s
MacBook Air *(Apple M1 chip, 8GB RAM, macOS 15.7.1)*				
msec/paper	14611.9ms	386.2ms	1363.4ms	16361.7ms
msec/citation	286.5ms	7.6ms	26.7ms	320.8ms
Total (sec)	1461.2s	38.6s	136.3s	1636.2s
WSL *(Ubuntu 22.04.4 LTS)*				
msec/paper	31970.5ms	329.4ms	2366.4ms	34666.5ms
msec/citation	626.9ms	6.5ms	46.4ms	679.7ms
Total (sec)	3197.1s	32.9s	236.6s	3466.7s

Table 3: Processing performance. We report the average time per paper and per citation in milliseconds.

readers to their study for a more detailed evaluation. In short, they characterize our tool as *"HalluCiteChecker generated fewer false positives but missed more problematic references."* This is because, for the reasons described in Section 3, our tool relies on closed databases and therefore does not support citations outside these databases. However, the databases used by our tool can be flexibly extended and supported in a modular manner. Accordingly, HalluCiteChecker provides an option to mark citations that are not covered by the selected databases as "not supported". Our YouTube video[3] demonstrates efficient detection of hallucinated citation candidates in real-world scenarios, further supporting its practical utility.

Efficiency. We evaluate efficiency by measuring the time required to verify citations in the first 100 long papers from ACL 2025. Experiments were conducted on three representative environments: MacBook Pro, MacBook Air (standard student laptop provided by our institution), and AMD Ryzen 7 PRO 6850U (2.70 GHz), 32GB RAM, Windows 11 Pro with WSL (Ubuntu 22.04.4 LTS). Table 3 shows that verification can be performed in at most 35 seconds per paper, and about 12 minutes for all 100 long papers. The primary bottleneck is the Citation Extractor component, indicating that further optimization of this stage would improve overall efficiency. Even on a MacBook Air, verification can be completed in approximately 15 seconds per paper, demonstrating the practical efficiency of the toolkit on commonly available hardware.

7 Conclusion

We introduced HALLUCITECHECKER, a toolkit for detecting hallucinated citations in scientific papers. We formalized citation verification as a structured task and presented a modular framework that decomposes the process into three subtasks, providing a clear foundation for future research. Moreover, as shown in Section 6, the toolkit enables fine-grained measurement of processing time for each component, facilitating systematic analysis and optimization in future development. Beyond its technical contributions, our goal is to support a more transparent and constructive scientific review process. By enabling systematic and reproducible citation verification, the toolkit can help reduce unintended desk rejections and unnecessary reviewer workload, thereby promoting responsible research practices among authors, reviewers, and organizers.

Limitations and future extension. It is designed as a transparent and extensible framework that facilitates future research. If perfect detection is needed, paid commercial tools or other more complex systems (Russinovich et al., 2026) may be more suitable. Rather, we envision HALLUCITECHECKER as a platform for community development and research on improving individual components, as well as a real-world benchmark for NER and information retrieval or for improving the entire pipeline. To facilitate such development, we prioritize a simple implementation as discussed in Section 3. Previous studies have also shown that citation errors can occur in author names, identifiers, URLs, and page numbers (Bianchi et al., 2025; Rao and Callison-Burch, 2026). Supporting the detection and verification of these types of errors is another direction for future work. Since the performance of individual modules also depends on the underlying libraries, their detailed evaluations can be found in the corresponding libraries. Finally, we welcome contributions to HALLUCITECHECKER!

References

2008–2026. Grobid. https://github.com/kermitt2/grobid.

2018–2024. Delft. https://github.com/kermitt2/delft.

Christoph Auer, Maksym Lysak, Ahmed Nassar, Michele Dolfi, Nikolaos Livathinos, Panos Vagenas, Cesar Berrospi Ramis, Matteo Omenetti, Fabian Lindlbauer, Kasper Dinkla, Lokesh Mishra, Yusik Kim, Shubham Gupta, Rafael Teixeira de Lima, Valery Weber, Lucas Morin, Ingmar Meijer, Viktor Kuropiatnyk, and Peter W. J. Staar. 2024. Docling technical report. *Preprint*, arXiv:2408.09869.

Max Bachmann. 2025. rapidfuzz/rapidfuzz: Release 3.13.0.

Fidan Badalova and Philipp Mayr. 2026. Detecting hallucinated and suspicious citations: What current tools can and cannot do. *Preprint*, arXiv:2607.22693.

Federico Bianchi, Yongchan Kwon, Zachary Izzo, Linjun Zhang, and James Zou. 2025. To err is human: Systematic quantification of errors in published ai papers via llm analysis. *Preprint*, arXiv:2512.05925.

Amanda Bienz, Carl Pearson, and Simon Garcia de Gonzalo. 2026. The case of the mysterious citations. *Preprint*, arXiv:2602.05867.

Jason P.C. Chiu and Eric Nichols. 2016. Named entity recognition with bidirectional LSTM-CNNs. *Transactions of the Association for Computational Linguistics*, 4:357–370.

Hiroyuki Deguchi, Yusuke Sakai, Hidetaka Kamigaito, and Taro Watanabe. 2024. mbrs: A library for minimum Bayes risk decoding. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 351–362, Miami, Florida, USA. Association for Computational Linguistics.

Jiazhou Ji and Xinru Lu. 2025. ReFLAIR: Enhancing multimodal reasoning via structured reflection and reward-guided learning. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 25401–25413, Suzhou, China. Association for Computational Linguistics.

Keigo Kusumegi, Xinyu Yang, Paul Ginsparg, Mathijs de Vaan, Toby Stuart, and Yian Yin. 2025. Scientific production in the era of large language models. *Science*, 390(6779):1240–1243.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 260–270, San Diego, California. Association for Computational Linguistics.

Quentin Lhoest, Albert Villanova del Moral, Yacine Jernite, Abhishek Thakur, Patrick von Platen, Suraj Patil, Julien Chaumond, Mariama Drame, Julien Plu, Lewis Tunstall, Joe Davison, Mario Šaško, Gunjan Chhablani, Bhavitvya Malik, Simon Brandeis, Teven Le Scao, Victor Sanh, Canwen Xu, Nicolas Patry, and 13 others. 2021. Datasets: A community library for natural language processing. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 175–184, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens, editors. 2026. *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, San Diego, California, United States.

Miryam Naddaf and Elizabeth Quill. 2026. Hallucinated citations are polluting the scientific literature. what can be done? *Nature*, 652(8108):26–29.

Bolaji David Oladokun, Rexwhite Tega Enakrire, Adefila Kolawole Emmanuel, Yusuf Ayodeji Ajani, and Adebowale Jeremy Adetayo. 2025. Hallucitation in scientific writing: Exploring evidence from chatgpt versions 3.5 and 4o in responses to selected questions in librarianship. *Journal of Web Librarianship*, 19(1):62–92.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

Delip Rao and Chris Callison-Burch. 2026. Bibtex citation errors in scientific publishing agents: Evaluation and mitigation. In *Third Conference on Language Modeling*.

David B. Resnik and Mohammad Hosseini. 2026. Hallucinated citations produced by generative artificial intelligence may constitute research misconduct when citations function as data in scholarly papers. *Accountability in Research*, 0(0):2645390. PMID: 41833014.

Mark Russinovich, Ram Shankar Siva Kumar, and Ahmed Salem. 2026. Phantom references: Hallucinated citations that survive peer review at top-tier conferences. *Preprint*, arXiv:2607.00738.

Yusuke Sakai, Hidetaka Kamigaito, and Taro Watanabe. 2026. HalluCitation matters: Revealing the impact of hallucinated references with 300 hallucinated papers in ACL conferences. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 47295–47376, San Diego, California, United States. Association for Computational Linguistics.

James H. Sweetland. 1989. Errors in bibliographic citations: A continuing problem. *The Library Quarterly: Information, Community, Policy*, 59(4):291–304.

William H. Walters and Esther Isabelle Wilder. 2023. Fabrication and errors in the bibliographic citations generated by chatgpt. *Scientific Reports*, 13(1):14045.

Chen Wang, Lai Wei, Yanzhi Zhang, Chenyang Shao, Zedong Dan, Weiran Huang, Ge Lan, and Yue Wang. 2026. Targeted exploration via unified entropy control for reinforcement learning. In *Findings of the Association for Computational Linguistics: ACL 2026*, pages 16789–16802, San Diego, California, United States. Association for Computational Linguistics.

Dashun Wang and Albert-László Barabási. 2021. *The Science of Science*. Cambridge University Press.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, and 3 others. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Zuyao Xu, Yuqi Qiu, Lu Sun, FaSheng Miao, Fubin Wu, Xinyi Wang, Xiang Li, Haozhe Lu, ZhengZe Zhang, Yuxin Hu, Jialu Li, Jin Luo, Feng Zhang, Rui Luo, Xinran Liu, Yingxian Li, and Jiaji Liu. 2026. Ghostcite: A large-scale analysis of citation validity in the age of large language models. *Preprint*, arXiv:2602.06718.

Chao Xue, Yao Wang, Mengqiao Liu, Di Liang, Xingsheng Han, Peiyang Liu, Xianjie Wu, Chenyao Lu, Lei Jiang, Yu Lu, Haibo Shi, Shuang Liang, Minlong Peng, and Flora D. Salim. 2026. Why supervised fine-tuning fails to learn: A systematic study of incomplete learning in large language models. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 30186–30213, San Diego, California, United States. Association for Computational Linguistics.

A Ethics and Broader Impact Statements

Responsible use and community impact. Our goal is not to discourage the use of AI-assisted research tools, but to provide a practical mechanism that enables authors, reviewers, and organizers to systematically verify citations and reduce unintended errors. This tool is intended to support a responsible research ecosystem by improving transparency, reducing unnecessary reviewer burden, and helping prevent avoidable desk rejections caused by citation errors. If tools such as HALLUCITECHECKER become widely adopted within the community, similar to existing tools such as ACL PUBCHECK, we encourage community-driven maintenance and governance to ensure transparency, sustainability, and alignment with community needs. We hope this work contributes to establishing fair, transparent, and responsible scientific practices. To ensure its long-term sustainability, we will maintain it and welcome contributions from the community, including new maintainers!

Social Impact for papers. We emphasize that the purpose of this task is to facilitate correction rather than punishment. Hallucinated citations can arise from a variety of sources, including contaminated bibliographic databases, unintended modifications introduced by paper management tools, and AI-assisted writing systems. As a result, they should not be assumed to reflect malicious intent or even the use of generative AI. We also advocate different handling depending on the publication stage. During peer review, hallucinated citations may be treated similarly to formatting violations, resulting in simple desk rejection for the current review cycle. For accepted or published papers, correction should generally be preferred over retraction, except in cases of clear intentional misconduct.

License. All tools, models, and datasets used in this work are distributed under permissive licenses, i.e., MIT License or Apache 2.0 License. Accordingly, we release HALLUCITECHECKER under the Apache 2.0 License to ensure broad accessibility, reproducibility, and responsible reuse. The papers used for evaluation and examples were obtained from the ACL Anthology, which is distributed under the Creative Commons Attribution 4.0 International License (CC BY 4.0). Therefore, their use in this study complies with applicable licensing terms and poses no legal or ethical concerns.

AI tool usage. This manuscript was prepared with the assistance of DeepL, Grammarly, and ChatGPT. The original content was written by the authors in Japanese, and these tools were used to produce and refine the English version. Most of the code was written from scratch manually. ChatGPT was occasionally used to search for code snippets and assist with code formatting. All content in the manuscript and the resulting code was verified by us, and we take full responsibility for this work.