ISNNI: 3007-3391 HEC Category: Y

Journal of Computers and Intelligent Systems
Vol. 4, No. 3, 2026, pages 381 - 392
Journal Homepage:
https://journals.iub.edu.pk/index.php/JCIS/

An Automated Recommendation System for Back-of-the-Book Index (BoBI)

Abdul Rehman Baloch, Yage Wang (王雅戈),
Library and Information Management, Suzhou University of Technology, Changshu, Jiangsu, P. R. China

ARTICLE INFO

Article History:

Received: July 07, 2026
Revised: August 05, 2026
Accepted: September22, 2026
Available Online: September30, 2026

Keywords:

Indexing System
Indexing Software
Automated Indexing
Index Generation
Back-of-the-Book index (BoBI)
Recommendation System

Classification Codes:

Funding:

This research received no specific grant from any funding agency in the public or not-for-profit sector.

ABSTRACT

The Indexing is the structured key terms and concepts within a document; it allows the readers to find particular information in the document quickly, efficiently, and correctly without needing to search the text as a whole. Existing indexing approaches face several limitations. It was done manually with the help of skilled indexers through well known applications, particularly when dealing with large information. Current systems used to select text one by one manually through software, or extract keywords by software, which often misses indexes keys already done by others and domain-specific terms. One promising direction involves the utilization of prior index datasets or libraries, where historical index entries are reused to improve efficiency and consistency in new documents. An interactive interface is required to allow indexers to review, accept, reject, or modify the recommended terms. To address these situations, this research proposes an Automated Index Recommendation System (AIRS) that streamlines the design book index through a structured three-phase approach, Input, Process, and Output. By bridging automation with indexer validation, the proposed system not only reduces workload, but also checks accuracy, and consistency. The proposed framework contributes to forward publishing and resource management by providing an efficient and reliable solution for modern needs. The results show a F1-score greater than 0.97, the filtering function redundant, the search accuracy 98%, and the HIR remained 98%, and the output achieved a precision of 99.5%.

OPEN ACCESS

© 2026 The authors published by JCIS. This is an Open Access Article under the Creative Common Attribution Non-Commercial 4.0

Corresponding Author's Email: *wyg@szut.edu.cn*

1. Introduction

The Book Indexing plays an important role in the organization of information. A back-of-the-book index (BoBI) provides structured references linking key terms and concepts to their locations within a document, enabling readers to efficiently access relevant information without browsing the entire text. The ability enhances the discover-ability, accessibility, and overall usability of complex information resources. Traditionally, Book indexing has been created manually by indexers, indexers carefully analysis, select important terms, and map them to sections and page numbers.

This manual approach is highly time consuming, costly, unscalable, and prone to inconsistencies, such as poor handling of synonyms hierarchical terms of the cross references "see" and "see also". With the

expansion of academic and professional content, manual methods are no longer sustainable, creating the need for automated solutions. In addition, most tools lack a robust index database management module, preventing intelligent handling of missing, redundant, or evolving keywords. Validation mechanisms are also weak, with limited support for expert-driven verification and refinement of automatically generated indexes.

However, one bright way refers to the use of prior index datasets or libraries, where historical index entries are reused to improve efficiency and consistency in new documents. The reuse approach is supported by international standards such as ISO 5963, Methods for analysing documents and selecting indexing terms [37]. Few solutions effectively reuse or match current documents with previously indexed data, leaving opportunities to improve efficiency and consistency unexplored.

The system should then recommend relevant index entries, ranked according to their importance, while also supporting hierarchical structures such as main entries, sub-entries, and cross-references.

The approach efficiently sort, organize text and numerical data according to predefined indexing rules such as eliminating identical rows of de-duplication, consolidates numerical values and merged these numerical values in ascending order, using commas for discrete numbers and hyphens to represent consecutive sequences (e.g., 1,3,5-7). It is supported by ISO 999 international standards of indexing process. aligned and incorporates human validation to preserve professional indexing judgment [36]. An interactive interface is required to allow indexers to review, accept, reject, or modify the recommended terms.

By bridging automation with indexer supervision and validation, the proposed system not only reduces workload, but also checks accuracy, and consistency. The proposed system contributes to forward publishing and resource management by providing an efficient and reliable solution for modern needs.

The aims and objectives of this research are to conduct a comprehensive and systematic literature review on indexing systems, to identify research gaps in past and present indexing approaches, to design a structured framework, to develop and implement the proposed system through algorithmic techniques using prior index datasets or libraries to support term to generate structured indexes comparable to manual indexing. To evaluate system accuracy against human aid indexes, state of the art methods.

2. Literature Review and Current Trands

2.1 Background of the Study

In this section, we discuss the background of the study and current trends in Back-of-the-Book Index (BoBI). The book index has advanced from manual to semi-automatic and automatic. Indexing software can automate tasks, entries, page referencing, and formatting. These programs can be separated into two categories. Standalone systems such as Cindex, Macrex and SkyIndex, and the second is embedded, like DEXembed [1] and WordEmbed [2], integrated within word processors.

Early word processors such as Wordperfect [3] and XyWrite [4] introduced manual index functions. Later, these lost importance due to the rise of more user-friendly software such as Microsoft Word [5]. Similarly, Nota Bene [6] extended XyWrite's capabilities by integrating bibliographic and multilingual support, targeting academic users.

During the 1980s to 1990s, Macrex [7], CINDEX [8], and SkyIndex [9] offered professional features, subheadings, formatting, and cross-references. These systems offer greater control compared to word processors. Other systems, including IndexAid [10], INDEXIT [11], Authex [12], and wINDEX [13], followed similar semi-automated approaches but varied in usability and adoption.

Alongside these, desktop based software like PageMaker [14], FrameMaker [15], and QuarkXPress [16] offered indexing as part of their capabilities. Although indexing in these systems remained mostly manual, they integrated indexing with design workflows, making them valuable in professional publishing contexts.

Later add-ins such as IXgen [17], emDEX [18], and IndexTension [19] specifically extended publishing platforms by embedding indexing functions.

In the past, experimentation with XML and other systems such as DocBook [20] and IndDoc [21], which combined manual tagging with computational support, was done. Fully automatic systems like TExtract [22] and later the PDF Index Generator [23] emerged, using keyword extraction to generate draft indexes that required indexer polish. These systems were balanced reflecting the continuing need for indexer oversight.

The VBAtechINDEXER [24] and iIndex Assistant [25] embedded indexing in word processors through macros and automation. Meanwhile, localized systems such as IndexStar [26] addressed linguistic challenges in Chinese indexing, showing regional adaptation of book index technologies. More softwares, such as the Text-to-Matrix Generator [27], apply text mining and vector-space models to indexing.

These are some of the softwares [7-11,13,25,26] that support the ISO 999 entry structure[36], alphabetical sorting, de-duplication, and page-range consolidation. But some of the software [1-6,14,15,17] provides indexing features, but does not formally claim ISO 999 compliance; In the end, some of the software [12,18-24,27] focuses on automatic extraction, keyword generation, rather than standard index construction [34].

2.2 Current Trends

Current trends in book indexing highlight the growing role of automation. One promising way involves the usage of index datasets or libraries, where historical index entries are reused to improve efficiency in new documents. However, the present systems still lack integrated systems.

Indexia [28, 29], IndexStudio [30], Sonar Bookends InDex Pro [31, 32] and IndexAI by Leanpub [33], AIDP [35] and others offered AI-based indexing that focuses on semantic understanding rather than rely less on historical datasets or established library indexing standards. These include semantic accuracy, ambiguous terms, and integration with publishing workflows.

Table 1. Comparison Table

Software Name	ISO 999[36]	ISO 5963 [37]
Cindex [8]	✓	✘
Macrex [7]	✓	✘
SkyIndex [9]	✓	✘
IndexAid [10]	✓	✘
INDEXIT [11]	✓	✘
iIndex Assistant [25]	✓	✘
IndexStar [26]	✓	✘

The comparison in Table 1 summarizes the selected indexing softwares in relation to ISO 999 [36] and ISO 5963 [37] standards. The assessment distinguishes explicitly documented standards compliance from partial functional support. The comparison highlights the differences in standards alignment among existing indexing tools and provides a basis for identifying opportunities for developing a standards-oriented automated indexing approach.

2.3 Challenges and Research Gap

Existing approaches face several limitations, either relying on manual, which is time-consuming, or on fully automated keyword extraction, which often misses indexes keys already done by others and domain-specific terms. One promising way involves the usage of index datasets or libraries, where historical index

entries are reused to improve efficiency in new documents. The solutions to reuse or match current documents with previously indexed data, leaving opportunities for improvement unexplored.

However, present systems still lack integrated systems, which could automate up to a good percentage of indexing tasks by suggesting or reusing validated terms. Moreover, validation workflows are often underdeveloped, offering limited support to automatically generated indexes. Moreover, most tools lack a robust index database, preventing the handling of missing index. Validation performance is also weak, with limited support and refinement of computer generated indexes. These gaps reveal the absence of automation, database-driven matching, and indexer validation to ensure accuracy for modern publishing needs.

Overall, the literature highlights a clear trajectory in book index from manual to standalone systems, then toward embedded and semi-automated. Advances in automation and the role of the indexer remain critical in ensuring contextual relevance and quality in the book indexing. Based on the literature review and the identified research gap , In the research framework, we define the aim and objectives of the research that will assist in the design, development, and evaluation of the proposed system.

Based on above challenges and research gap and the need to achieve accuracy in modern indexing systems. The proposed system is aligned with ISO 999 [36], and for reuse of prior index datasets is an approach supported by international standards such as ISO 5963, which incorporates indexer validation and professional indexing judgment.

3. Research Methodology

In the methodology, we will discuss the requirements of the system, the design of the system, and the proposed algorithm. Here we describe the datasets along with preparation and experimentation. Finally, present the results and performance of the evaluation of the proposed approach.

3.1 Requirement Engineering

The proposed system requires a clear definition of both functional and non-functional requirements to ensure the system achieves its intended goals. The functional requirements include the ability to upload and process the documents, the system should perform text preprocessing tasks including tokenization, stemming, and stopword removal.

Based on processed data, the system must identify potential index terms through frequency based extraction, semantic analysis, and machine learning approaches. In the above approaches, the system recommends relevant index entries, ranked according to their importance, then indexers to review, accept, reject, or modify the recommended terms. Non-functional requirements are equally critical to system efficiency and acceptance. Performance should be optimized so that publication is processed. The system must be scalable enough to handle large academic books. Usability is also a requirement; therefore, the interface should be designed to be accessible.

Accuracy is the goal of achieving at least a good percentage compared to human-created indexes. Portability should also be ensured so that the system operates on different platforms. Finally, the system should be able to incorporate updates in indexing rules as future trends evolve. In addition, the system has to have some constraints in addition to the functional and non-functional requirements. The system must be able to integrate well with the publishing tools and must also have online and offline options for the various types of users. Furthermore, the system supports multilingual indexing to address global publishing needs and provides custom export formats that are tailored to the requirements of different publishers.

3.2 Design Engineering

The proposed Automated Index Recommendations System (AIRS) is designed to transform the traditional process. The system architecture is divided into three main phases, Input, Process, and Output, each addressing specific tasks to generating indexes. Fig.1 is defined in the following.

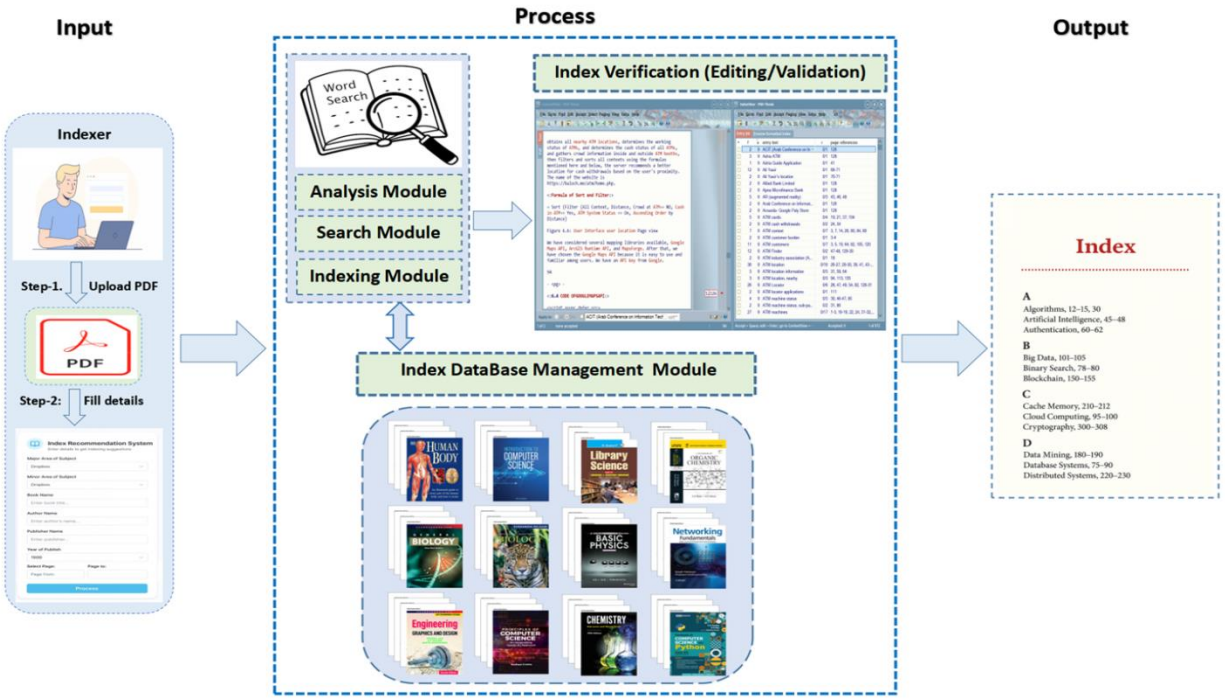


Figure. 1. System Architecture

3.3 The Sequence of the Methods

To design an AIRS system and its components, the following sequences of methods are to be followed as defined in Fig: 2.

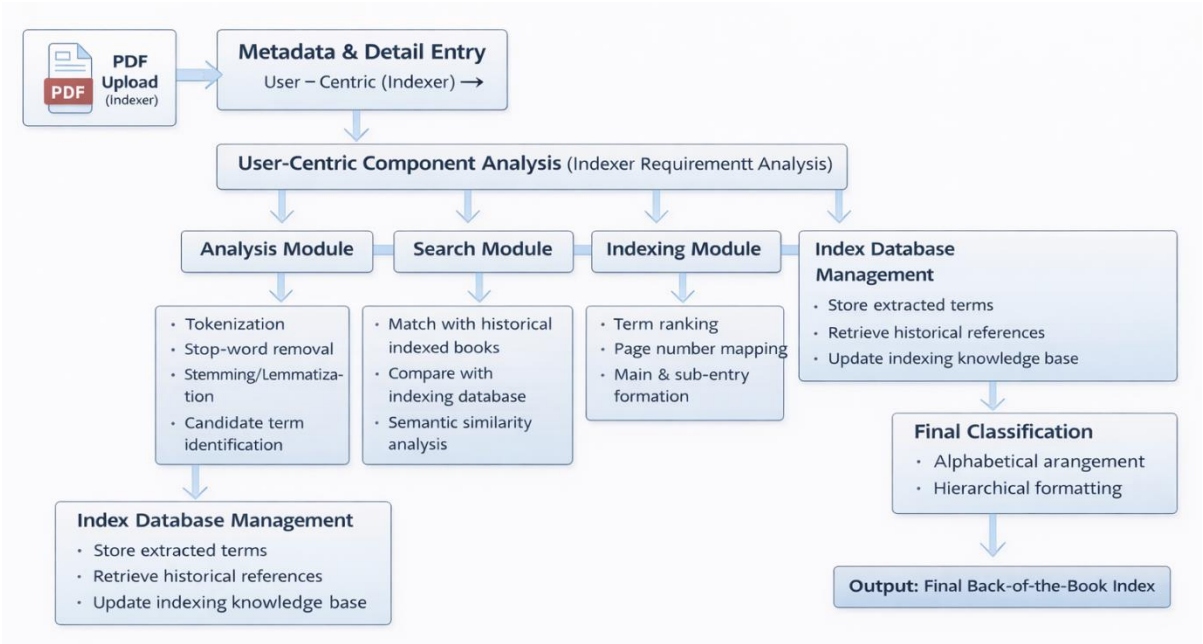


Figure 2. The Sequence of the Methods for the AIRS

3.4 The Algorithm of the System

The Input Phase is the foundation stage; it begins with the indexer's role in providing three primary inputs, the PDF document D, descriptive metadata $M = \{m_1, m_2, \ldots, m_k\}$, and the range of defined pages $R = [p_s, p_e]$, and to generate an organized alphabetical output index representing I.

Index coverage and the indexing boundary to provide some information about documents, including the major area of subject (e.g., Engineering, Medical, Science), the minor area of the subject (e.g., Civil Engineering), the title, author, publisher, and year of publication to the contextual identification.

Additionally, PDF documents are represented as finite set of page ranges $D = \{p_1, p_2, p_3, \ldots, p_n\}$, where each p represents the document page and n shows the total number of pages, and since the indexer specifies a page range (e.g. pages 9 to 70 of a total of 78), only pages are considered for indexing, excluding the front matter, content, or reference sections, these ranges of pages will represent with R, start page Ps to the end page Pe, the subset D_R is an operational data set for further analysis, the documents are defined as:

$$D = \{p_1, p_2, p_3, \ldots, p_n\} \quad (1)$$

$$D_R = \{p_i \in D \mid p_s \leq i \leq p_e\}$$

$$\text{thus,} \quad D_R \subseteq D \quad (2)$$

Once this information is completed, the system transitions to the next stage by initiating the process phase. In index database management, to improve indexing coherence to support the reuse of previously used index words, the system incorporates a historical index database, h_k presents a validated entry of the previously used index, t_j for each candidate term, the reuse condition using similarity of the threshold θ, the update index term set integrating both reused and newly validated terms $I_1 = R \cup (T_f - R)$ defined as.

$$H = \{h_1, h_2, h_3, \ldots, h_r\} \quad (3)$$

$$\mathrm{Sim}(t_j, h_k) \quad (4)$$

$$R = \{t_j \in T_f \mid \exists h_k \in H : \mathrm{Sim}(t_j, h_k) \geq \theta\} \quad (5)$$

$$I_1 = R \cup (T_f - R) \quad (6)$$

However, the database is updated using $H \leftarrow H \cup I_1$, allowing continuous learning. This mechanism improves the terminology of indexing and the overall quality of indexing documents.

$$H \leftarrow H \cup I_1 \quad (7)$$

The automated indexing, professional end of the book indexing required indexer expertise, then a verification function performed as $I_v = V(I_1)$, and after validated sorted index entries $I = \mathrm{Sort}_\alpha(I_v)$, final index represented as $I = \{(t_j, O(t_j))\}$ and return I.

$$I_v = V(I_1) \quad (8)$$

$$I = \mathrm{Sort}_\alpha(I_v) \quad (9)$$

$$I = \{(t_j, O(t_j))\} \quad (10)$$

Finally, the third sub-process, Index Verification, enables indexer oversight, editing, and validation by the indexer. These areas can manually add missing terms and remove irrelevant entries. Once this perfection is complete, the index is validated by automated integration with human oversight and the database management module effectively delivers high-quality index.

Algorithm 1: An AIRS Document

Input: Document D, Metadata
$M = \{m1, m2, \ldots, mk\}$,
Page Range $R = [ps, pe]$
Output: Structured Alphabetical Index I
Initialization:
1-Define document: $D = \{p_1, p_2, p_3, \ldots, p_n\}$
2-Restrict pages: $D_R = \{p_i \in D \mid p_s \leq i \leq p_e\}$
(a) Analysis Module:
3-Extract terms: $T = \text{ExtractTerms}(D_R)$, $T = \{t_1, t_2, t_3, \ldots, t_m\}$
4-Apply filtering: $T_f = \text{F}(T)$
(b) Search Module:
5-Compute mapping: $O(t_j) = \{p_i \in D_R \mid t_j \in p_i\}$
6-Construct index: $I_0 = \{(t_j, O(t_j)) \mid t_j \in T_f\}$
(c) Index Database Management:
7-Historical database: $H = \{h_1, h_2, h_3, \ldots, h_r\}$
8-Compute similarity: $\text{Sim}(t_j, h_k)$
9-Reuse: $R = \{t_j \in T_f \mid \exists h_k \in H : \text{Sim}(t_j, h_k) \geq \theta\}$
10-Updated index set: $I_1 = R \cup (T_f - R)$
11-Store in database: $H \leftarrow H \cup I_1$
(d) Verification Module:
12-Human validation: $I_v = \text{V}(I_1)$
where V represents indexer editing, approval, and refinement

(e) Output:
13-Alphabetical ordering: $I = \text{Sort}_\alpha(I_v)$
14-Final structured index: $I = \{(t_j, O(t_j))\}$ *(alphabetically ordered)*
Return I

The final phase of the output of the AIRS will deliver an alphabetically arranged index, containing keywords alongside their page references. Manages not only the data extracted, but also the repository of previously generated index records. When processing a new document, the system compares extracted keywords with prior entries in the database since relevant terms overlapping directly match historical data. Reuse of indexing knowledge accelerates the process, promotes uniformity across related publications, and reduces redundancy in manual work. The proposed approach sorting and organizing textual and numerical data to predefined rules.

4. Experimental Results and Analysis

The experimental results and analysis section presents the implementation and experimental setup of the proposed framework. It describes experimental evaluation through the use of Python, including data set selection, preprocessing, world extraction, optimization, and index generation. The experiments were conducted on a controlled dataset.

The proposed AIRS algorithm was implemented using an experiment that was evaluated using standard documents. Since there was no publicly available platform for systematic evaluation of back-of-the-book index generation, a controlled dataset was generally constructed for this study. Three self-made documents were created to simulate the book. Each document contained 600 and 3100 lines of text and included multiple heading levels, duplicate terms, nested sub-sections, and cross-references.

The documents were classified based on simplicity, just like Doc-1 is simple with basic structure, Doc-2 is medium, with multi-level structure, and Doc-3 is complex with deep nesting and varied formatting structure. Each document processed through the AIRS modules, analysis, search, index database management, verification, and index output. This disciplined experimental design allowed for systematic validation of the algorithm's performance.

4.1 Evaluation Metrics

The algorithm of the recommendation system was evaluated using the quantitative metrics of the term extraction score, F_1 = 2 × (Precision × Recall) / (Precision + Recall), the precision of occurrence, the range of page R, the hierarchical Integrity, and with the reuse of similarity, Percentage of words reused represent with $\mathrm{Sim}(t_j, h_k) \geq \theta$, and in the end manual correction entries represent with V(I), the final index with I and the total execution time represent with (ms) runtime.

def generate_index(document, metadata, page_range, historical_terms): ps, pe = page_range

- **Analysis:** terms = extract_terms(document, ps, pe) filtered_terms = filter_terms(terms, metadata)
- **Search:** preliminary_index = map_occurrences(document, filtered_terms)
- **Database Management:** updated_index = reuse_terms(preliminary_index, historical_terms)
- **Verification:** validated_index = human_validation (updated_index)
- **Output:** final_index = alphabetical_sort (validated_index)
- return final_index

4.2 Result analysis.

The results anaylsis section, it is essential to explain the scope and significance of the recommendation system, highlighting its findings. The primary goal to assist indexers in generating relevant indexes, which traditionally has been manual process. book indexes are necessary for improving the accessibility of content, allowing readers to promptly locate specific topics. The system is designed to reduce these effort.

The evaluation of system performed using multiple quantitative metrics, including the term extraction score (F1-score), precision of occurrence, page range (R), hierarchical integrity, percentage of words reused, and manual correction entries V(I). These metrics provide a comprehensive assessment of the system's performance, the F1-score captures the balance between precision and recall, offering insights into the system's ability to identify relevant index. The precision of occurrence further ensures that terms suggested by the system accurately reflect the occurrences within the document, while page range assessment measures the coverage and relevance of the term across the text.

The reuse of similarity metric shows the level to which AIRS is utilizing already indexed material to obtain consistency and minimize redundancy, which is a valuable attribute when indexing several documents on the same topic or collection. The manual correction entries V(I) offer a complementary perspective by quantifying the human effort required to refine the generated index. The resulting index I is the product of the system output, which is based on both the automated recommendations and the human corrections and hence, it gives a realistic value of the utility of a system in the professional world.

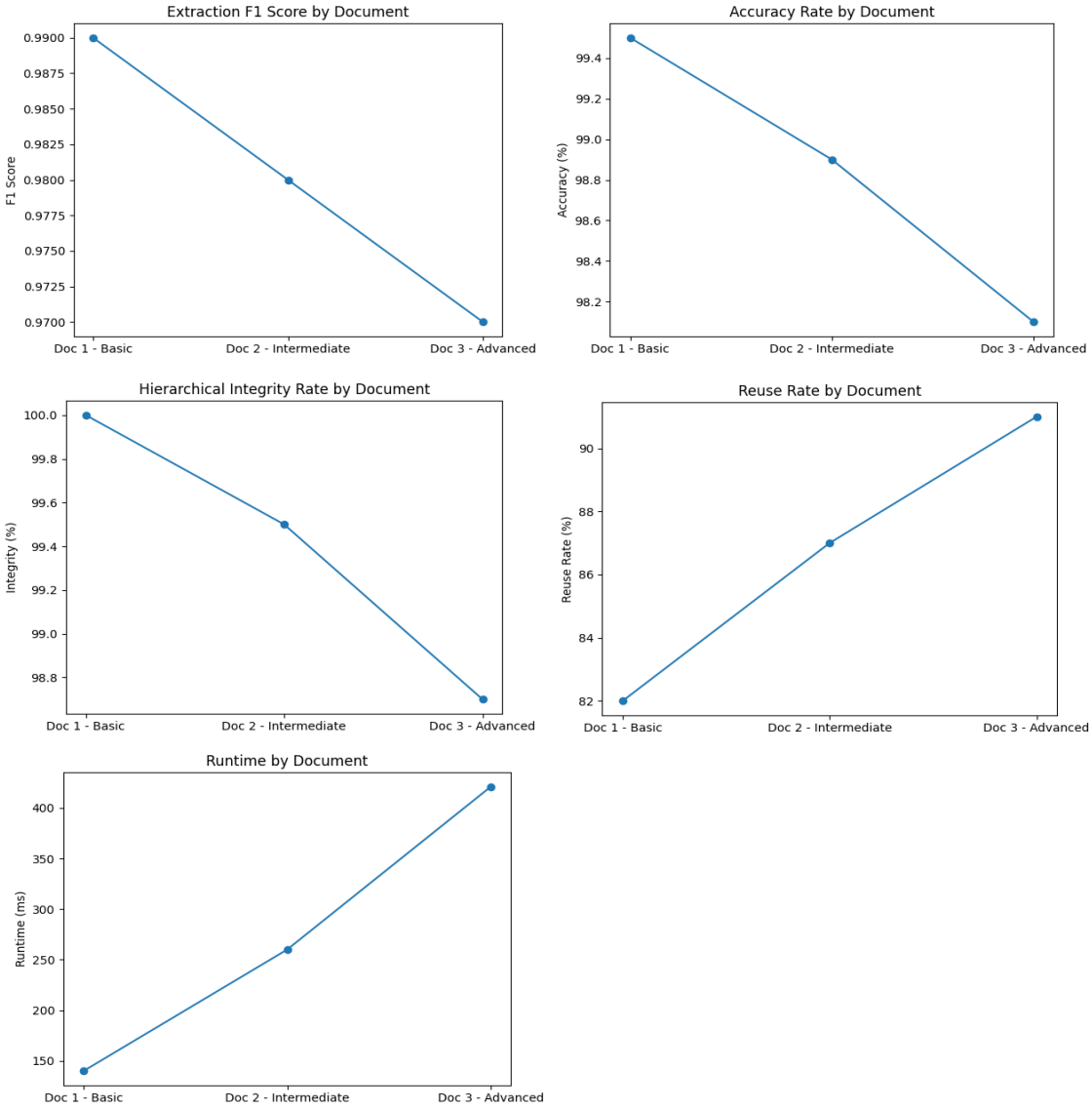


Figure 3. The results of Extraction F1, Accuracy rate, Hierarchical Integrity rate, Resue rate and Runtime

In addition to accuracy, execution time (ms) was evaluated to ensure the system can efficiently for longer documents. Overall, the results confirm that AIRS provides a practical, efficient, and accurate solution for book indexing.

As graphs generated through code included to visually represent the experimental performance of system. These graphical results illustrate evaluation metrics and behavior in different documents mentioned in Fig. 3.

The results showing good execution, grip F1 scores above 0.97, filtering function and searching accuracy function exceeding 98%, HIR remained 98%. The database management of the index module was improved through similarity reuse. In the end verification module, the reduced indexer workload with manual refinement is low. The output structure achieved 100% character set ordering accuracy, ensuring professional formatting quality.

Table 1. Experimental Results of AIRS

Document	Unit	F1	Accuracy	H-I Rate	Reuse	Ordering	Runtime (ms)
1 -Basic	620	0.99	99.5 %	100 %	99.5 %	100 %	140
2-Intermediate	1050	0.98	98.9 %	99.5 %	99.5 %	100 %	260
3- Advanced	3100	0.97	98.1 %	98.7 %	99.5 %	100 %	421

4.3 Comparative Analysis

The trend of book indexing has been growing towards AI-automation. Such systems like MACREX, CINDEX, SkyIndex, IndexAid, Authex, DocBook, IndDoc, Indexia, IndexStudio, Sonar Bookends InDex Pro, and IndexAI emphasize understanding and automated suggestion generation.

Table: 2. Comarative Summary integrity across Indexing systems.

Model / System	Hierarchical Integrity	Evaluation Basis
Proposed AIRSModel	99.5%	Experimental evaluation
Indexia [28]	Not reported	Literature review
IndexStudio [30]	Not reported	Literature review
InDex Pro [31]	Not reported	Literature review
IndexAI [33]	Not reported	Literature review

These are based on contextual analysis rather than historical libraries or datasets. Although the proposed approach indexing model outperforms contemporary in quantitative evaluation. With an average of 0.98, and recall of 0.98, in addition, it maintains HI at approximately achieves good accuracy.

Although other provide friendly interfaces, In contrast, the proposed approach integrates morphological layers, resulting in improved domain alignment. Overall, the comparative results suggest that the current system represents an important step toward automation, making it more suitable for scalable and academically rigorous automated book indexing applications.

The AIRS evaluation focuses on assessing the proposed framework with human-generated indexes, and existing indexing software, Direct comparative experiments against others extraction were not conducted. Therefore, the reported performance metrics demonstrate AIRS performance within the selected dataset rather than alternative approaches. Comparative evaluation using standardized datasets and human-generated indexing outputs is an important direction for future research.

5. Discussion and Conclusions

In the Discussion and conclusions, the Indexing allows the readers to find particular information in the document quickly and efficiently without needing to search the text as a whole. Existing systems face several limitations. The Proposed systems step up to the promising direction of the utilization prior libraries, where historical index are reused to improve in new documents.

The proposed framework provides automation with an expert validation approach, agile processing, and structured output; it significantly reduces the quality of manual index generation. At the same time, it ensures accuracy and consistency, thereby contributing to the advancement of information retrieval and back-of-the book index creation. The proposed indexing system effectively overcomes the traditional indexing by combining database driven matching with automation, and indexer validation. Its three phase framework supports efficient document handling, while allowing for the refinement and validation of missed entries. The results show that the proposed system achieves the good performance mentioned above.

In addition, the conclusion index output achieves about 99.5% functional accuracy, indicating the reliability of the indexing process. By bridging agile automation with indexer supervision, the AIRS framework offers an efficient solution for publishing and document management environments.

Data Availability

The data used in this research was developed as an experimental standard for evaluating the performance of the proposed system. To ensure transparency and reproducibility of the research, the data, results, figures, code and related materials used in this study are publicly available in the following repository. https://github.com/arehman-baloch/AIRS-System.git

Author Contributions: Abdul Rehman Baloch: Conceptualization, Methodology, Software, Data curation, Writing Original draft preparation. **Yage Wang（王雅戈）**: Supervision.

Acknowledgements: The author expresses gratitude to the Talented Young Scientist Program, China (TYSP, Pakistan-24-026), and Suzhou University of Technology, Jiangsu Province, P.R. China for providing research facilities, support for valuable guidance, and collaboration during the research.

Funding: The authors receive no funding.

AI Assistance Disclosure: The authors declare that generative artificial intelligence tools were used during the preparation of this work for language formatting and language clarity to improvements of content to readability for academic tone, and diagram, code generation.

Compliance with Ethical Standards: It is declared that all authors do not have any conflict of interest. In addition, informed consent was obtained from all individual participants included in the study.

6. References

[1] L. Haskins, "DEXembed, WordEmbed, and IndexDeconstructor: How do they work and why would I want to use them?," in Proc. San Diego Conf., San Diego, CA, USA, 2012.

[2] J. A. Lamb, *WordEmbed v3.11 User Manual*, 2015. [Online]. Available: https://jameslambsite.wordpress.com/wp-content/uploads/2015/09/wordembed-v3-112.pdf

[3] K. Van Winkle, *WordPerfect 6.0 for DOS Bible*, IDG Books Worldwide, Foster City, CA, USA, 1994.

[4] Improved XyWrite IV, *PC Mag.*, vol. 9, no. 3, p. 54, Feb. 13, 1990.

[5] Microsoft Corp., *Microsoft Word* [Computer software], 1983.

[6] Nota Bene, "Nota Bene official website," 2025. [Online]. Available: https://www.notabene.com/[Accessed: Aug. 14, 2025].

[7] G. Beare and D. Calvert, "MACREX- A history," *The Indexer*, vol. 24, no. 1, pp. 18-20, 2004.

[8] OpenCindex, "OpenCindex: Open source indexing software," 2025. [Online]. Available: https://www.opencindex.org/ [Accessed: Aug. 12, 2025].

[9] Sky Software, "About us," 2025. [Online]. Available: https://skyits.com/about-us/ [Accessed: Aug. 13, 2025].

[10] A. N. Grosch, "Index-Aid: Computer-assisted back-of-the-book indexing," *The Electron. Libr.*, vol. 4, no. 5, pp. 278-280, 1986.

[11] L. K. Fetters, "INDEXIT: An economical, but limited indexing program," *Database*, vol. 9, no. 5, pp. 54-56, Oct. 1986.

[12] Authex: Printed index production, *Inf. Technol. J.*, vol. 9, no. 6, pp. 451-453, 1985, doi: 10.1108/EB024197.

[13] K. Zakaria, "Back of the book index1," presentation, Scribd, 2009. [Online]. Available: https://www.scribd.com/presentation/22000634/Back-of-the-Book-Index1

[14] T. Köklü, "Adobe PageMaker: Unveiling the legacy of desktop publishing," *Designer Info*, 2023. [Online]. Available: https://www.designer-info.com/adobe-pagemaker/

[15] D. J. Murray, "Frame technology and FrameMaker," *IEEE Ann. Hist. Comput.*, vol. 41, no. 3, pp. 62-74, Jul.-Sep. 2019, https://doi.org/10.1109/MAHC.2019.2926662

[16] Quark, “Creative software solutions,” 2025. [Online]. Available: https://www.quark.com/
[17] L. Haskins, “Using IXgen to manage markers,” 1999. [Online]. Available: https://www.luciehaskins.com/resources/IXgenManagingMarkersArticle.pdf
[18] Leximation, Inc., “emDEX: Back-of-the-book indexing add-in for Adobe FrameMaker,” 2008. [Online]. Available: https://leximation.com/toolsearch/?dev=EMD
[19] Codesco, “Indextension: The innovative solution for industrial extension,” 2023. [Online]. Available: https://www.codesco.com/en/indextension.html [Accessed: Aug. 31, 2025].
[20] J. Kosek, “Mastering DocBook indexes,” *XML.com*, 2004. [Online]. Available: https://www.xml.com/pub/a/2004/07/14/dbndx.html
[21] H. Zargayouna, T. A. E. Mekki, L. Audibert, and A. Nazarenko, “IndDoc: An aid for the back-of-the-book indexer,” *The Indexer*, vol. 25, no. 2, pp. 122-125, 2006.
[22] R. Bennett, “TExtract: A regular user’s view,” *The Indexer*, vol. 27, no. 2, pp. 84-87, 2009, https://doi.org/10.3828/indexer.2009.26
[23] PDF Index Generator, “PDF Index Generator,” 2025. [Online]. Available: https://pdfindexgenerator.com/ [Accessed: Aug. 26, 2025].
[24] VBA Technology, “VBA Indexer,” 2007. [Online]. Available: http://vbatech.com/indexer/
[25] iIndex, “iIndex software,” 2025. [Online]. Available: https://www.iindex.com/index.php [Accessed: Aug. 16, 2025].
[26] Y. Wang and G. Wang, “Development and indexing of ‘Index Stars’,” in *Proc. Dictionary and Digitization Seminar*, 2004.
[27] K. Koutropoulou and E. Gallopoulos, “TMG-BoBI: Generating back-of-the-book indexes with the Text-to-Matrix-Generator,” in *Proc. 10th Int. Conf. Inf., Intell., Syst. Appl. (IISA)*, Patras, Greece, 2019, pp. 1-8, https://doi: 10.1109/IISA.2019.8900745.
[28] Indexia, “AI book indexing tool-Create professional back-of-the-book indexes,” 2026. [Online]. Available: https://www.indexia.tech/ [Accessed: Feb. 5, 2026].
[29] Indexia Team, “Index Locorum: AI-powered citation indexing for classics and legal scholarship,” Feb. 3, 2026. [Online]. Available: https://www.indexia.tech/blog/index-locorum-citation-indexing [Accessed: Feb. 5, 2026].
[30] IndexStudio, “IndexStudio AI-powered book indexing,” 2026. [Online]. Available: https://indexstudio.app/ [Accessed: Feb. 5, 2026].
[31] Virginia Systems Software Services Inc., “Sonar Bookends InDex Pro,” 2026. [Online]. Available: http://www.virginiasystems.com/ [Accessed: Feb. 5, 2026].
[32] Virginia Systems Software Services Inc., “Automatic index and table of contents generators for Adobe InDesign,” 2026. [Online]. Available: http://www.virginiasystems.com/products_d.html#indesigndemo [Accessed: Feb. 5, 2026].
[33] Leanpub, “IndexAI: Create an index for your book with one click,” 2026. [Online]. Available: https://leanpub.com/services/index_ai [Accessed: Feb. 5, 2026].
[34] Abdul Rehman Baloch and Wang Yage, “The Evolution of Back-of-the-Book Index (BoBI) Systems: A Review”, ASSAJ, vol. 5, no. 2, pp. 1151－1173, May 2026. https://assajournal.com/index.php/36/article/view/1776
[35] S. Ahmed and W. Yage, “AIDP: A Standards Driven Human-Automation Architecture for Intelligent and Secure Document Indexing,” Dialogue Social Science Review (DSSR), vol. 3, no. 12, pp. 7－17, Dec. 2025
[36] International Organization for Standardization. (1996). Information and documentation-Guidelines for the content, organization and presentation of indexes (ISO 999:1996). https://www.iso.org/standard/5446.html
[37] International Organization for Standardization. (1985). Documentation-Methods for examining documents, determining their subjects, and selecting indexing terms (ISO Standard No. 5963:1985). https://www.iso.org/standard/12158.html