Available online at www.ijournalse.org

Emerging Science Journal

(ISSN: 2610-9182)

Vol. 10, No. 1, February, 2026

Structure-Aware Chunking for Complex Tables in Retrieval-Augmented Generation Systems

Xin-Kuang Koay [1, 2], Lee-Yeng Ong [1, 2*], Pey-Yun Goh [1, 2]

[1] *Faculty of Information Science and Technology, Multimedia University, Malacca 75450, Malaysia.*

[2] *Centre for Advanced Analytics, CoE for Artificial Intelligence, Faculty of Information Science and Technology, Multimedia University, Melaka 75450, Malaysia.*

Abstract

Retrieval-Augmented Generation (RAG) is a hybrid method that combines information retrieval with large language models to generate context-aware, factually grounded responses. However, the RAG system relies heavily on well-structured input data to generate accurate and contextually relevant responses. Documents with complex table layouts pose significant challenges, as most chunking strategies are text-centric and often overlook table-rich documents containing multi-column and multi-row structures. Hence, this study proposes a customized structure-aware chunking framework specifically designed for university course documents containing multi-column, multi-row tables with nested headers. The framework employs Camelot for high-fidelity table extraction, followed by customized logic that constructs semantically coherent chunks by preserving academic term, subject name, credit hour, and category. This prevents semantic fragmentation during retrieval. The proposed method is evaluated using the RAGAS framework and compared against several baselines using standard parsing and chunking techniques. Results show that the proposed approach achieves the highest answer accuracy of 0.73 and substantially improves retrieval relevance and contextual precision. These findings demonstrate the framework's effectiveness in handling structure-dependent academic queries. This study highlights that ensuring both parsing quality and chunking strategy is essential to retain semantic relationships in table-rich documents, offering a practical improvement for RAG systems in structurally complex scenarios.

Keywords:

Retrieval-Augmented Generation;

Large Language Model;

Chunking;

Table Parsing;

Complex Table Layout.

Article History:

Received:	12	June	2025
Revised:	09	December	2025
Accepted:	13	January	2026
Published:	01	February	2026

1- Introduction

Retrieval-augmented generation (RAG) is a hybrid method combining the generative capabilities of large language models (LLMs) with external document retrieval mechanisms to produce contextually relevant and up-to-date responses [1]. By leveraging retrieved passages during the generation phase, RAG helps to reduce hallucination and improve factual accuracy, making them particularly effective for tasks that require access to external knowledge [1-4]. RAG has proven state-of-the-art performance in open-domain question answering [5, 6], where models are required to retrieve and synthesize information from a broad corpus. RAG has also proven effective in knowledge-intensive dialogue, where generating accurate responses depends on dynamically sourced facts [5, 7]. In addition, RAG has been applied successfully to document summarization, where the integration of external evidence leads to more faithful and comprehensive summaries [8].

While previous studies on RAG have focused on enhancing RAG frameworks on unstructured textual data [9], limited attention has been given to documents that contain complex tabular formats. Some recent work has explored table-based retrieval, but such studies often assume direct access to pre-structured data sources such as relational databases [10, 11],

* **CONTACT**: lyong@mmu.edu.my

DOI: http://dx.doi.org/10.28991/ESJ-2026-010-01-09

bypassing the parsing challenges in PDFs associated with real-world documents. In contrast, institutional datasets such as university course structures are typically embedded in PDF files featuring irregular layouts, merged cells, and nested headers. As shown in our proposed dataset (refer to Figure 5), these complex tables encode semantic relationships through visual structure rather than linear text, making it difficult to extract meaningful units for retrieval. This presents a unique challenge in preserving term-subject-category-credit hour relationships when performing parsing and chunking.

A key factor affecting the performance of RAG systems is the quality of the input document, particularly how it is parsed and chunked [2, 12, 13]. Although some researchers have proposed advanced parsers such as LlamaParse for complex tables [14], it still often struggles to accurately capture all relevant details in our proposed dataset, which features highly intricate layouts even though it performs well in other studies [15, 16]. Other parsing strategies, such as Tabula, perform well on simple tables but lack robustness when handling nested or merged structures. Tools like pdfplumber and PyMuPDF are optimized for general text extraction rather than table parsing [17].

In addition, most existing chunking methods rely on fixed-token or recursive splitting techniques, which are designed for linear, unstructured documents such as articles. These methods do not account for the spatial and semantic relationships inherent in complex tables, often splitting logically connected rows, columns, or headers across separate chunks [18]. Such fragmentation degrades retrieval accuracy and leads to hallucinations or incoherent responses in downstream generation. To our knowledge, no prior work has proposed an end-to-end framework that begins with PDF-based table parsing and proceeds to customized chunking while explicitly preserving semantic relationships across term, subject, category, and credit hour dimensions.

To address these challenges, a customized structure-aware chunking framework is proposed to improve semantic coherence in downstream retrieval. This study focuses specifically on tabular documents derived from university course structure PDFs, which feature nested headers, multi-row and multi-column layouts, merged cells, and alternating column semantics. The proposed framework integrates Camelot for high-fidelity table extraction, preserving the original grid alignment of merged headers, row hierarchies, and term-column associations. On this basis, a structure-aware chunking strategy is applied to semantically group subject names, credit hours, and categories under their respective academic terms. Each chunk is then enriched with structured metadata to enhance indexing and retrieval efficiency. The key contributions of this work are as follows:

- A customized structure-aware chunking strategy is introduced to mitigate semantic fragmentation, which often occurs when complex tabular structures are split by recursive or fixed-token chunking methods.
- The proposed chunking framework is integrated with Camelot-based high-fidelity table parsing and semantic retrieval to construct a RAG pipeline capable of preserving contextual integrity in semantically dense academic documents.
- A comprehensive evaluation using the RAGAS framework is conducted, demonstrating significant improvements in content relevance, retrieval precision, and answer accuracy when compared across multiple established baselines.

The organization of this paper is as follows. Section 2 reviews existing studies on RAG systems, with a focus on parsing and chunking strategies for complex documents. Section 3 describes the proposed structure-aware chunking framework, including the Camelot-based parsing method and the semantic chunk construction process. Section 4 presents the experimental setup and evaluation metrics used to benchmark the framework. Section 5 discusses the results, compares them with baseline models, and highlights key findings and limitations. Section 6 concludes the study and outlines potential directions for future work.

2- Literature Reviews

2-1- Retrieval-Augmented Generation (RAG)

Retrieval-Augmented Generation (RAG) is a hybrid framework introduced by Lewis et al. [1]. RAG integrates large language models (LLMs) with external document retrieval to generate responses that are both factually accurate and contextually relevant. RAG bridges the gap between pre-trained LLMs and external knowledge sources, allowing models to incorporate up-to-date or domain-specific information that goes beyond their original training corpus. RAG enhances the ability of LLMs by retrieving relevant document chunks from external knowledge bases through semantic similarity calculations [19].

The RAG pipeline is built around three primary components, which are Indexing, Retriever, and Generator. As shown in Figure 1, the Indexing component follows a four-step process to prepare raw documents for semantic retrieval [2, 8, 19]. First, source documents are parsed and pre-processed to extract clean textual content. Next, the content is segmented into smaller, manageable chunks to optimize retrieval accuracy and efficiency. These chunks are then transformed into high-dimensional vector representations using embedding models. The resulting vectors are stored in a vector database (or vector store), forming the basis for similarity-based retrieval during inference. The Retriever uses this vector index to identify the most relevant chunks in response to a user query, while the Generator synthesizes a definitive answer by combining the query with the retrieved content.

Although the RAG system performs well with unstructured data, such as narrative text, it still encounters significant limitations when processing documents with complex tabular structures. Traditional chunking strategies, such as the fixed-token, recursive, and sliding window methods, are optimized for linear text and agnostic to spatial or tabular layout [20-24]. When applied to complex tables with nested headers, multi-row, and multi-column layouts, these methods often disrupt semantic coherence by splitting logically related rows, columns or merged cells across multiple chunks.

Recent research has attempted to address this issue by introducing advanced table parsers [2, 25] to enhance preprocessing. However, these solutions often flatten the visual structure of tables during extraction, resulting in the loss of critical contextual relationships, such as column pairings, category groupings, and hierarchical headers [26]. This fragmentation significantly degrades the quality of retrieval and the accuracy of generated responses. Despite advancements in table parsing, there is still a lack of structure-aware chunking methods designed specifically to preserve the semantic integrity of complex tabular data within RAG pipelines.

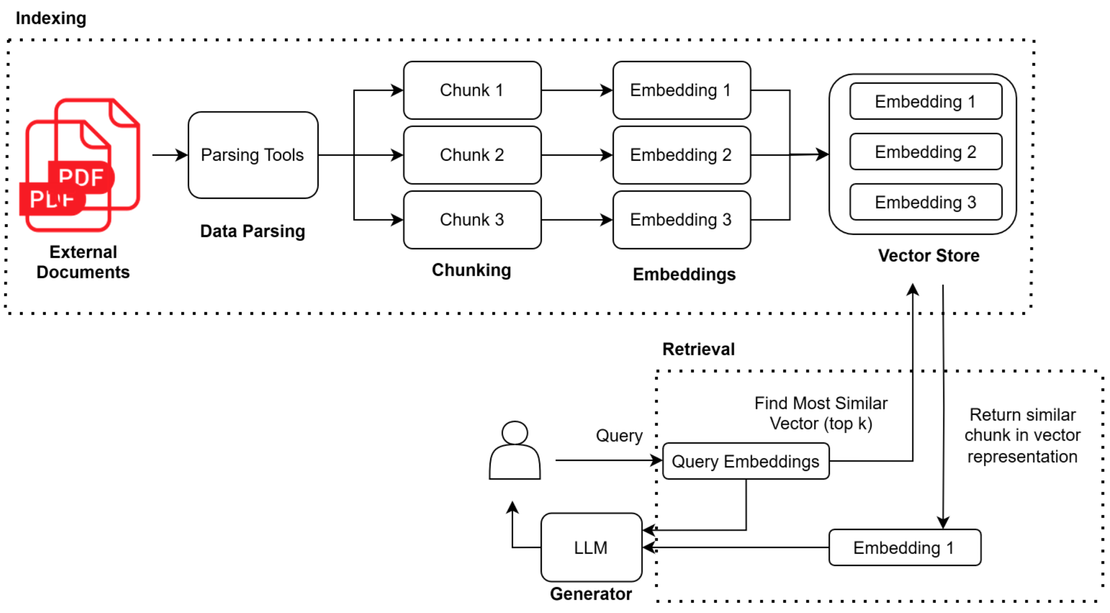
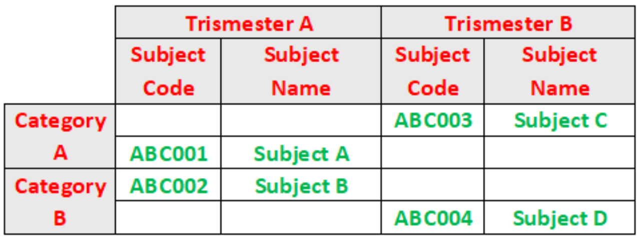


Figure 1. General Pipeline of RAG

2-2- Data Parsing on Table and Chunking

Figure 2 illustrates an example of complex tabular layouts found in structured academic documents. This table features nested headers (e.g., Trimester A and Trimester B, each with its own sub-headers like "Subject Code" and "Subject Name"), as well as multi-row labels on the left (e.g., Category A and Category B), which significantly complicate automated data extraction. The red text in the figure highlights multi-level column and row headers, while the green text illustrates cell values that are contextually linked across rows and columns. For example, "ABC003 – Subject C" belongs to Category A under Trimester B, even though the row begins empty, meaning the category label is implied from above. Each logical entry notably spans two adjacent cells, with one holding the primary content and the other extending its meaning or completing the data unit. Some cells are left blank to indicate implied values from previous rows or columns. Such structural intricacies are difficult for most existing parsers to handle, particularly traditional parsers [27-29]. The table features complex column and row-header configurations, as well as multi-line rows, which often lead to flattened or misaligned extraction. As a result, tools fail to capture the correct associations between subjects, categories, and trimesters, especially when used for retrieval in downstream tasks.

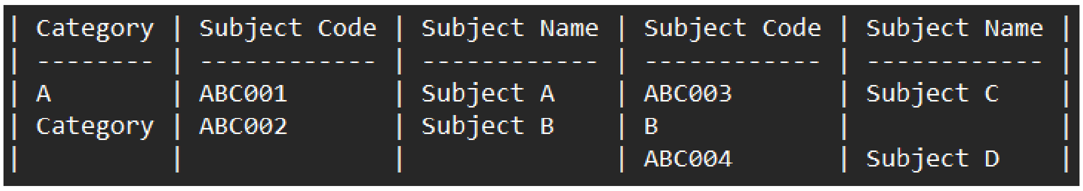


	Trismester A		Trismester B	
	Subject Code	Subject Name	Subject Code	Subject Name
Category A			ABC003	Subject C
	ABC001	Subject A		
Category B	ABC002	Subject B		
			ABC004	Subject D

Figure 2. Sample layout of a complex table with multi-row and multi-column format

Although the complex table layout is easily understood by human readers, conventional PDF parsers such as PyPDFLoader often struggle with positional inaccuracy during parsing [17]. While advanced parsers such as LlamaParse [30, 31] can successfully extract tables, they still face challenges in accurately preserving structure when handling highly complex features like nested headers and multi-row or multi-column formats. LlamaParse is considered a powerful table parser when enabled in Markdown mode that specifically designed for complex table layouts. Despite being set to output in Markdown mode, LlamaParse encounters difficulties in accurately parsing such tables [16]. This is because LlamaParse often linearizes or flattens the information in tables, resulting in the loss of positional relationships between headers and values. These structural complexities pose major challenges for parsing methods, which flatten tables during extraction.

An example of such flawed output is shown in Figure 3, where LlamaParse processes the complex table from Figure 2. The output fails to preserve the nested header structure and maintain the correct positional alignment of cell values. For example, Trimester A, Trimester B, and Category B disappear entirely, leaving disconnected subject codes and names without context. Blank cells that are critical for implied groupings are either dropped or misaligned, leading to incorrect subject associations. This demonstrates how even advanced parsers fail to preserve the nuanced structure of complex academic tables, leading to data loss and misinterpretation in downstream analysis.



```
| Category | Subject Code | Subject Name | Subject Code | Subject Name |
| -------- | ------------ | ------------ | ------------ | ------------ |
| A        | ABC001       | Subject A    | ABC003       | Subject C    |
| Category | ABC002       | Subject B    | B            |              |
|          |              |              | ABC004       | Subject D    |
```

Figure 3. Sample extracted output using LlamaParse

Even when table parsing is successful, the effectiveness of a RAG system is fundamentally limited by how well the parsed data is segmented into chunks [2, 13]. Existing chunking strategies such as fixed-token segmentation and recursive chunking are primarily designed for linear text. These methods divide text based on predefined token limits or syntactic patterns, without considering spatial relationships, visual alignment, or underlying structural dependencies [21]. While these techniques perform adequately for unstructured documents, they are not well suited to handling complex tables [3]. Complex tables are often fragmented by traditional chunking strategies that split semantically related elements such as rows of associated values or header-cell relationships across different chunks. This fragmentation disrupts the logical flow of information and makes it more difficult to retrieve relevant content effectively.

3- Proposed Framework

The proposed framework is grounded in the theoretical assumption that accurate parsing is a prerequisite for effective semantic chunking, especially when dealing with complex tabular documents. Our method builds upon the strong structural reconstruction capability of Camelot. Its lattice mode ensures that grid-based tables are parsed with high fidelity, without data loss or misalignment. This reliable parsing output forms the basis for our customized structure-aware chunking method, which groups subjects, credit hours, and categories according to the spatial and semantic relationships present in the table layout.

Parsing and chunking are not independent stages. They form an interdependent pipeline in table-based RAG systems [2, 12, 13]. If the parsing stage fails to preserve critical structural elements such as merged headers, row spans, or alternating columns, any downstream chunking logic will produce fragmented or misleading outputs. For this reason, our approach adopts a tightly coupled design. Accurate parsing enables targeted and structure-preserving chunking strategies that match the document's inherent layout.

To preserve the semantic and structural integrity of layout-rich academic tables, the proposed framework integrates Camelot parsing with customized chunking logic. These two components work together to enhance the relevance of retrieved context. The complete framework is illustrated in Figure 4. It begins with table indexing, where blue and green headers represent columns and rows. These indexed tables are parsed using Camelot, which generates structured representations of the raw tabular data. The chunking workflow consists of four key modules: (1) Term Identification and Extraction, (2) Subject-to-Term Mapping, (3) Category Carry-Forward Logic and Credit Hour Validation, and (4) Structured Triple Construction. These processed outputs are embedded and stored in a vector database, which serves as the input context for the LLM during response generation. Each phase is explained in the following sections.

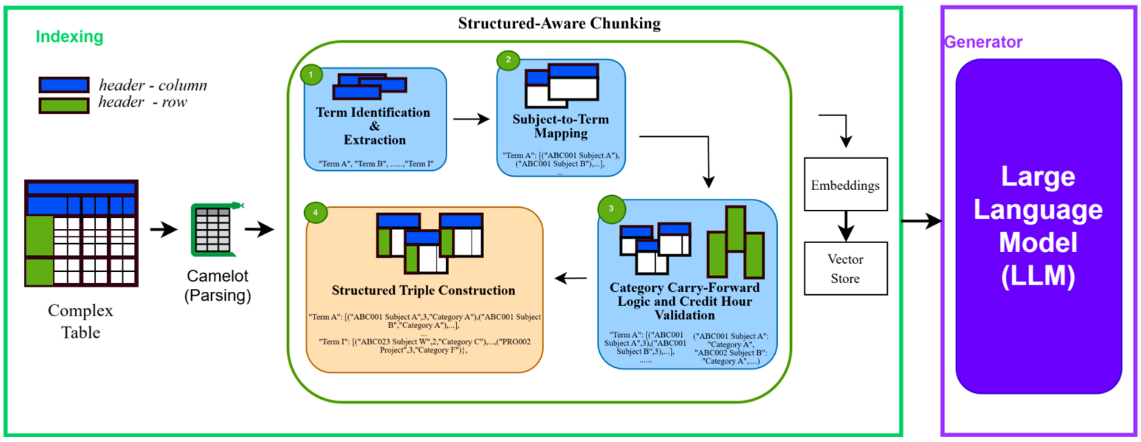

Indexing
header - column
header - row
Complex Table
Camelot (Parsing)
Structured-Aware Chunking
1
Term Identification & Extraction
"Term A", "Term B", ...,"Term I"
2
Subject-to-Term Mapping
"Term A": [("ABC001 Subject A"), ("ABC001 Subject B"),...],
...
3
Category Carry-Forward Logic and Credit Hour Validation
"Term A": [("ABC001 Subject A",3),("ABC001 Subject B",3),...],
......
("ABC001 Subject A": "Category A", "ABC002 Subject B": "Category A",....)
4
Structured Triple Construction
"Term A": [("ABC001 Subject A",3,"Category A"),("ABC001 Subject B","Category A"),...],
...
"Term I": [("ABC023 Subject W",2,"Category C"),...,("PRO002 Project",3,"Category F")},
Embeddings
Vector Store
Generator
Large Language Model (LLM)

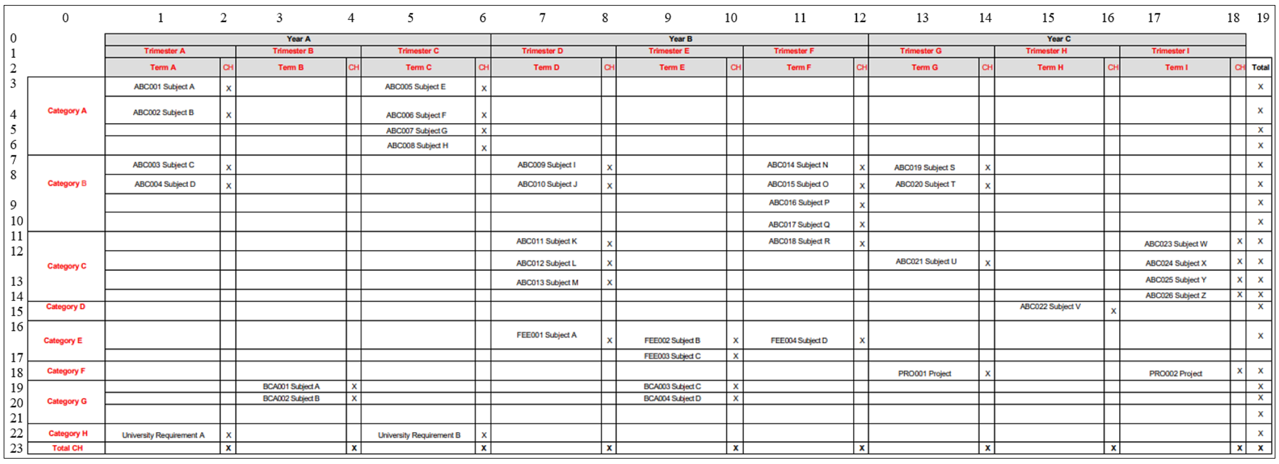
Figure 4. Proposed Framework

3-1- Dataset

This study uses a dataset consisting of internal academic documents from a university. Specifically, it uses course structure tables that outline subject offerings across academic terms, as illustrated in Figure 5. Due to institutional privacy constraints, the full content of these documents cannot be disclosed publicly. However, the table's layout and structural characteristics are critical to the methodology, as the layout conveys implicit information such as term grouping, course categorization and subject availability through visual formatting elements like merged cells, alignment and spacing.

Each document contains one or more large tables arranged in a grid format. Columns represent academic terms (e.g., Term A, Term B, etc.), and rows are grouped by subject category (e.g., Category A, Category B, etc.). Each course entry typically spans a pair of adjacent columns. The odd-numbered column (e.g., 1, 3, 5, …) contains the course title along with its associated term, while the even-numbered column (e.g., 2, 4, 6, …) immediately to the right holds the corresponding credit hour (CH), which belongs to the course on its left. This alternating pattern is applied consistently across all terms. Header labels for terms and trimesters are often merged across multiple columns to indicate hierarchical relationships across academic years. Similarly, category labels are vertically aligned and span several rows, signifying that all subjects listed underneath belong to the same academic group.

Empty cells are used to denote the absence of a course offering in a particular term. In some cases, credit hour values are omitted but implied through carry-forward logic based on the context of the row or column. Visual formatting, such as alignment, spacing and line breaks, plays a vital role in preserving these semantic relationships, yet this information is often lost or 'flattened' during conventional parsing. Consequently, these layout-specific cues present significant challenges for traditional PDF parsers and rule-based chunking methods. A structure-aware approach is required that can dynamically interpret the positional logic embedded in the document design.

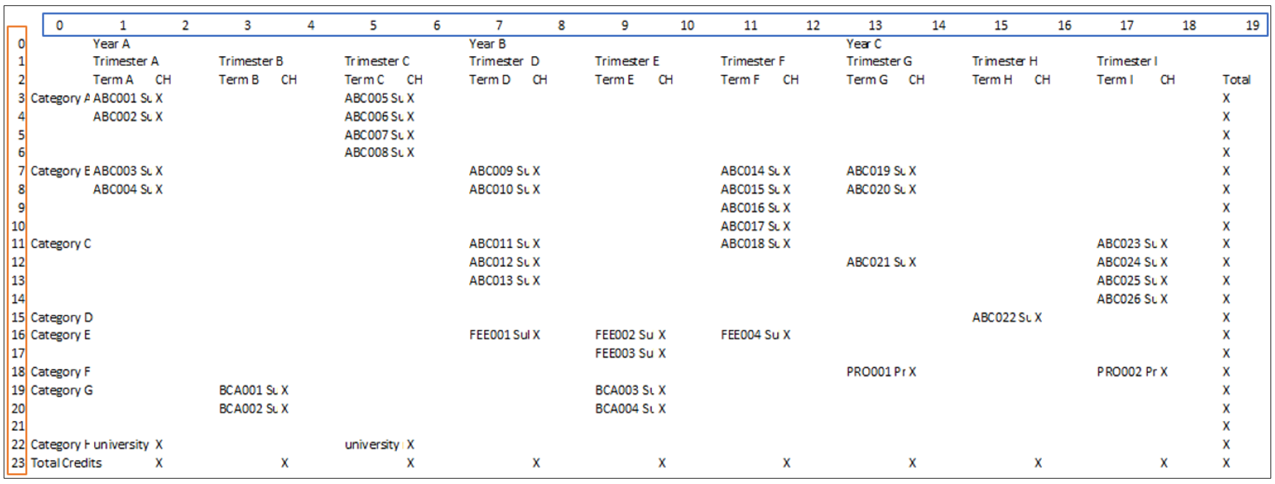


	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
0		Year A						Year B						Year C						
1		Trimester A		Trimester B		Trimester C		Trimester D		Trimester E		Trimester F		Trimester G		Trimester H		Trimester I		
2		Term A	CH	Term B	CH	Term C	CH	Term D	CH	Term E	CH	Term F	CH	Term G	CH	Term H	CH	Term I	CH	Total
3		ABC001 Subject A	X			ABC005 Subject E	X													X
4	Category A	ABC002 Subject B	X			ABC006 Subject F	X													X
5						ABC007 Subject G	X													X
6						ABC008 Subject H	X													X
7		ABC003 Subject C	X					ABC009 Subject I	X			ABC014 Subject N	X	ABC019 Subject S	X					X
8	Category B	ABC004 Subject D	X					ABC010 Subject J	X			ABC015 Subject O	X	ABC020 Subject T	X					X
9												ABC016 Subject P	X							X
10												ABC017 Subject Q	X							X
11								ABC011 Subject K	X			ABC018 Subject R	X					ABC023 Subject W	X	X
12	Category C							ABC012 Subject L	X					ABC021 Subject U	X			ABC024 Subject X	X	X
13								ABC013 Subject M	X									ABC025 Subject Y	X	X
14																		ABC026 Subject Z	X	X
15	Category D															ABC022 Subject V	X			X
16	Category E							FEE001 Subject A	X	FEE002 Subject B	X	FEE004 Subject D	X							X
17										FEE003 Subject C	X									
18	Category F													PRO001 Project	X			PRO002 Project	X	X
19				BCA001 Subject A	X					BCA003 Subject C	X									X
20	Category G			BCA002 Subject B	X					BCA004 Subject D	X									X
21																				X
22	Category H	University Requirement A	X			University Requirement B	X													X
23	Total CH		X		X		X		X		X		X		X		X		X	X

Figure 5. Course Structure Table

3-2- Parsing

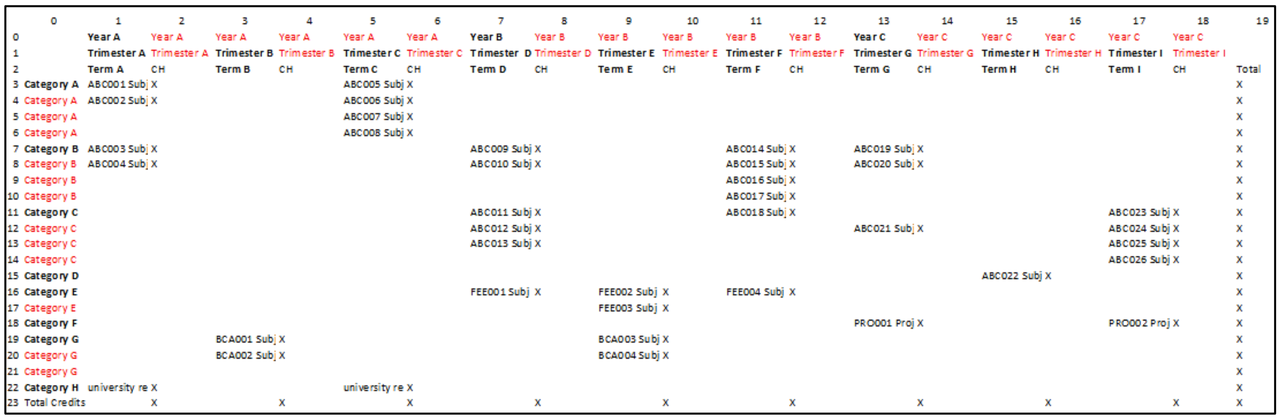
In this study, Camelot is employed as a Python-based table extraction library due to its demonstrated effectiveness in accurately parsing structured, grid-like tables. Recent research has shown that Camelot performs better in table detection compared to alternatives such as Tabula, pdfplumber, and PyMuPDF [17]. Studies highlight that Camelot consistently outperforms Tabula across various document types for table detection tasks, while pdfplumber and PyMuPDF are more suitable for general text parsing rather than structured table extraction. Compared to these tools, Camelot's lattice mode enables high-fidelity table extraction by detecting explicit cell borders, merged cells, and aligned headers. In contrast, Tabula is more effective for simpler tables and often struggles with multi-column or nested layouts. Camelot's ability to accurately identify individual table elements, even when they are visually merged or wrapped, makes it particularly well-suited for complex tabular documents where headers and labels often span multiple rows or columns, as illustrated in Figure 6.



	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
0		Year A						Year B						Year C						
1		Trimester A		Trimester B		Trimester C		Trimester D		Trimester E		Trimester F		Trimester G		Trimester H		Trimester I		
2		Term A	CH	Term B	CH	Term C	CH	Term D	CH	Term E	CH	Term F	CH	Term G	CH	Term H	CH	Term I	CH	Total
3	Category A	ABC001 Su	X			ABC005 Su	X													X
4		ABC002 Su	X			ABC006 Su	X													X
5						ABC007 Su	X													X
6						ABC008 Su	X													X
7	Category B	ABC003 Su	X					ABC009 Su	X			ABC014 Su	X	ABC019 Su	X					X
8		ABC004 Su	X					ABC010 Su	X			ABC015 Su	X	ABC020 Su	X					X
9												ABC016 Su	X							X
10												ABC017 Su	X							X
11	Category C							ABC011 Su	X			ABC018 Su	X					ABC023 Su	X	X
12								ABC012 Su	X					ABC021 Su	X			ABC024 Su	X	X
13								ABC013 Su	X									ABC025 Su	X	X
14																		ABC026 Su	X	X
15	Category D															ABC022 Su	X			X
16	Category E							FEE001 Sul	X	FEE002 Su	X	FEE004 Su	X							X
17										FEE003 Su	X									X
18	Category F													PRO001 Pr	X			PRO002 Pr	X	X
19	Category G			BCA001 Su	X					BCA003 Su	X									X
20				BCA002 Su	X					BCA004 Su	X									X
21																				X
22	Category H	university	X			university	X													X
23	Total Credits		X		X		X		X		X		X		X		X		X	X

Figure 6. **Camelot's "lattice" mode**

Next, the "copy text" option in Camelot is employed to automatically fill blank cells by copying text horizontally or vertically until a new text value appears. The output is illustrated in Figure 7, where the bold text represents the original headers, and the red text shows the auto-filled values produced by the copy-and-fill operation. This option is crucial for preserving both horizontal and vertical text alignments during parsing. It ensures that content is not misaligned or incorrectly grouped when headers span multiple rows or columns. For instance, the "Category" labels originally merged across several rows are automatically repeated for each relevant row, allowing course entries to be correctly associated with their corresponding category. However, subject titles and credit hour entries are not repeated because they are structured as distinct, term-specific cell pairs and are not visually merged in the original table layout. As a result, Camelot preserves them in their original positions without auto-filling, ensuring that only true header values are propagated. By enabling this feature, the logic applied in the subsequent chunking stage can accurately interpret each cell based on its row and column position, thereby preserving the structural integrity of the original table.



	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
0		Year A	Year A	Year A	Year A	Year A	Year A	Year B	Year B	Year B	Year B	Year B	Year B	Year C	Year C	Year C	Year C	Year C	Year C	
1		Trimester A	Trimester A	Trimester B	Trimester B	Trimester C	Trimester C	Trimester D	Trimester D	Trimester E	Trimester E	Trimester F	Trimester F	Trimester G	Trimester G	Trimester H	Trimester H	Trimester I	Trimester I	
2		Term A	CH	Term B	CH	Term C	CH	Term D	CH	Term E	CH	Term F	CH	Term G	CH	Term H	CH	Term I	CH	Total
3	Category A	ABC001 Subj	X			ABC005 Subj	X													X
4	Category A	ABC002 Subj	X			ABC006 Subj	X													X
5	Category A					ABC007 Subj	X													X
6	Category A					ABC008 Subj	X													X
7	Category B	ABC003 Subj	X					ABC009 Subj	X			ABC014 Subj	X	ABC019 Subj	X					X
8	Category B	ABC004 Subj	X					ABC010 Subj	X			ABC015 Subj	X	ABC020 Subj	X					X
9	Category B											ABC016 Subj	X							X
10	Category B											ABC017 Subj	X							X
11	Category C							ABC011 Subj	X			ABC018 Subj	X					ABC023 Subj	X	X
12	Category C							ABC012 Subj	X					ABC021 Subj	X			ABC024 Subj	X	X
13	Category C							ABC013 Subj	X									ABC025 Subj	X	X
14	Category C																	ABC026 Subj	X	X
15	Category D															ABC022 Subj	X			X
16	Category E							FEE001 Subj	X	FEE002 Subj	X	FEE004 Subj	X							X
17	Category E									FEE003 Subj	X									X
18	Category F													PRO001 Proj	X			PRO002 Proj	X	X
19	Category G			BCA001 Subj	X					BCA003 Subj	X									X
20	Category G			BCA002 Subj	X					BCA004 Subj	X									X
21	Category G																			X
22	Category H	university re	X			university re	X													X
23	Total Credits		X		X		X		X		X		X		X		X		X	X

Figure 7. **Camelot's "copy text" option**

3-3- Structured-Aware Chunking

The core contribution of this study lies in the Structure-Aware Chunking strategy. Structure-Aware Chunking is specifically customized to match the layout and content arrangement of university course structure tables. The primary goal of our approach is to preserve the semantic relationships embedded within the table format, especially the associations between academic terms, subjects, credit hours, and categories. Preserving these relationships is essential for enabling the RAG system to answer fundamental academic queries such as "What subjects are offered in Term A?".

To support this goal, each generated chunk is structured to include a complete and logically connected set of information. For instance, a single chunk may consist of one academic term, all subjects offered during that term, and the corresponding credit hours and categories for each subject. This format ensures that each chunk contains internally consistent and meaningful data, which improves retrieval precision during inference.

In contrast to traditional chunking methods that often break related items into separate chunks, the structure-aware approach guarantees that each chunk reflects a well-defined semantic unit. As a result, the RAG system can retrieve fewer but more accurate chunks that directly support the user's question. This design reduces retrieval noise and improves the factual grounding of generated responses. The effectiveness of this design can be clearly observed in Table 6, which provides a side-by-side comparison of chunk outputs. It is evident that the structure-aware chunks contain well-formed, query-relevant data, which enables the RAG system to correctly answer even basic academic questions. This is an outcome that traditional chunking methods frequently fail to deliver due to fragmented retrieval.

The Structure-Aware Chunking process involves four sequential steps, each carefully aligned with the original table structure to preserve subject-term relationships, categories, and credit hours. To enhance clarity, each step can refer back to Figure 7 to facilitate understanding of how the Structure-Aware chunking strategy operates on a complex academic table. A summary of the intermediate outputs at each stage is presented in Table 1.

Table 1. Intermediate Outputs at Each Step of the Structure-Aware Chunking

Step	Temporary Store	Output
1	-	“Term A”, “Term B”, ..., “Term I”
2	-	“Term A”: [("ABC001 Subject A"), ("ABC002 Subject B"), …..], “Term B”:[("BCA001 Subject A"),…],….
3	{"ABC001 Subject A": "Category A", "ABC002 Subject B": "Category A", ….}	“Term A”: [("ABC001 Subject A",3), ("ABC002 Subject B", 3) …..], “Term B”:["BCA001 Subject A, 3",…],….
4	-	“Term A”: [("ABC001 Subject A", 3, "Category A"), ("ABC002 Subject B", 3, "Category A") …..], “Term B”:[("BCA001 Subject A", 3, "Category A"), …],….

1. *Term Identification and Extraction*

The chunking process begins by identifying academic terms (e.g., Term A, Term B, ..., Term I) from the parsed table. Since Camelot’s lattice mode maintains the original grid layout, the header row containing term labels can be directly extracted. These term labels serve as primary grouping keys, which will be used to organise the corresponding course data into structured chunks in the following steps. For example, columns 2–3 represent Term A, columns 4–5 represent Term B, etc.

2. *Subject to Term Mapping*

Once the term headers have been identified, the table is processed row by row. For each row, course titles are read from the odd-numbered columns, starting from the first column of each term. An entry is considered a valid subject name if the cell is non-empty and contains a recognizable course code. Using the column index, each course is assigned to its corresponding term based on the headers extracted in Step 1. At this stage, a mapping of course titles to terms is created, although credit hour and category information are not yet included. This step establishes the base structure for each academic term.

For instance, in Term A (column 2), ABC001 Subject A and ABC002 Subject B are read from column 2. If the cell is not empty, the subject is mapped to Term A. This results in a dictionary such as: "Term A": ["ABC001 Subject A", "ABC002 Subject B", etc.]. The process is repeated iteratively for each row and term column until all valid course codes are mapped to their respective terms.

3. *Category Carry Forward Logic and Credit Hour Validation*

Although the “copy text” option in Camelot is enabled to maintain header alignment, a carry-forward strategy is also applied as an additional safeguard. In this approach, the first column of each row represents the subject category (e.g., "Category A"). However, due to the use of merged cells in many tables, category labels may not be explicitly repeated in every row. To address this, the most recently identified category is carried forward and applied to subsequent rows until a new category appears. For example, if Category A is found in row 2, it is assumed to apply to all subsequent courses in that block unless a new label (e.g., Category B) is detected. This ensures that each subject is classified consistently and correctly, maintaining semantic integrity across the dataset.

At this stage, the result is a subject-to-category dictionary, such as: {"ABC001 Subject A": "Category A", "ABC002 Subject B": "Category A", etc.}. The assigned categories are temporarily stored and later combined with subject titles and credit hours in final step to form complete, semantically coherent triples. The term-to-subject mapping established in Step 2 (e.g., "Term A": ["ABC001 Subject A", "ABC002 Subject B"]) remains intact and is not modified by this step.

Step 3 serves to independently enrich the data with category information without altering the term-based structure. The final combination of subject title, credit hour, category, and academic term occurs in Step 5, where all prior mappings are merged into structured triples for downstream retrieval.

Each course title is followed by its credit hour in the adjacent column (even-numbered index). The parser pairs course names with their corresponding credit hours only if the value is valid (i.e., a numeric digit). Invalid or empty entries are ignored. For example, if ABC001 Subject A appears in column 2, and column 3 contains a numeric value such as 3, the pair is considered valid. Otherwise, it is skipped. This ensures that each subject is accurately linked to a quantitative workload, which is essential for academic planning and reliable RAG-based generation. Additionally, in cases where a row is entirely empty or contains multiple critical blanks (e.g., missing both subject title and credit hour), the framework is designed to skip such rows entirely to prevent the risk of incorrect mappings. These validation checks act as safeguards to ensure that only complete and contextually meaningful entries are included in the final structured triples. As a result, the system maintains high semantic integrity and avoids propagating misclassifications due to parsing anomalies or table irregularities. In this stage, the result is a dictionary such as: "Term A": [("ABC001 Subject A", 3), ("ABC002 Subject B", 3), etc.].

4. Structured Triple Construction

In the final stage, each course entry is combined into a structured triple consisting of the subject title, its corresponding credit hour, and the associated category. These components, previously extracted through subject-to-term mapping (Step 2), category assignment and credit hour validation (Step 3), are now merged to form semantically coherent units. For instance, a course such as ABC001 Subject A with 3 credit hours under Category A would be represented as the triple ("ABC001 Subject A", 3, "Category A").

These triples are then grouped according to their respective academic terms, producing a structured, dictionary-like format that preserves the layout of the original table. This organization ensures that each subject is contextually aligned with its term, credit weight, and classification, maintaining both the semantic and structural integrity of the document. The resulting term-level chunks serve as the final units for embedding, indexing, and retrieval in the downstream RAG process.

4- Experiment Setup

4-1- Evaluation Metrics

To evaluate the effectiveness of the proposed chunking strategy, the RAGAS (Retrieval-Augmented Generation Assessment) framework proposed by Es et al. [32] is employed. RAGAS offers a standardized approach to assess the quality of responses generated by a RAG system, using a combination of automatic metrics and model-assisted judgments. Six RAGAS metrics, shown in Equations 1 through 7, are used to assess system performance.

Faithfulness measures the factual consistency between the generated answer and the retrieved context. It is calculated using Equation (1), where |V| represents the number of statements that are supported according to the LLM, and |S| is the total number of statements. A higher faithfulness score indicates that the generated response is more accurately grounded in the retrieved documents.

$$F = \frac{|V|}{|S|} \tag{1}$$

Answer Relevancy evaluates how well the generated answer addresses the user's question. Answer Relevancy is computed using Equation 2, where n is the number of generated questions q_i, $\text{sim}(q, q_i)$ is cosine similarity between the embeddings of q and q_i. A higher Answer Relevancy score indicates that the generated content remains closely aligned with the user's query, reflecting semantic relevance.

$$AR = \frac{1}{n}\sum_{i=1}^{n} \text{sim}(q, q_i) \tag{2}$$

Content Relevance measures semantic similarity between the generated and ground truth answers. To estimate context relevance score, Equation 3 is used, where the number of extracted sentences divided by total number of sentences in c_q , where c_q represent the context provided for the question q. A higher Content Relevance score indicates that the generated answer effectively incorporates information from the retrieved context, reflecting strong semantic alignment.

$$AR = \frac{number\ of\ extracted\ sentences}{total\ number\ of\ sentences\ in\ c_q} \tag{3}$$

Answer Accuracy measures the agreement between the model's generated response and the human-annotated ground truth for a given question. The evaluation involves two judges rating the response, with the ratings converted into a normalized [0, 1] scale. The final score is the average of both judgments. Higher scores indicate that the model's answer closely matches the reference in terms of factual correctness and completeness.

$$F1\ score = \frac{|TP|}{(|TP|+0.5*(|FP|+|FN|))} \tag{4}$$

Content Precision is a metric that measures the proportion of relevant chunks in the retrieved contexts. It is calculated as the mean of the precision@k for each chunk in the context. precision@k is the ratio of the number of relevant chunks at rank k to the total number of chunks at rank k. Higher Content Precision indicates that the system is retrieving more contextually relevant information, improving the grounding quality for response generation.

$$CP = \frac{\sum_{k=1}^{K}(precision@k*v_k)}{total\ number\ of\ relevant\ items\ in\ top\ k} \tag{5}$$

$$Precision@k = \frac{TP@k}{(TP@k+FP@k)} \tag{6}$$

Context Recall measures how many of the relevant documents (or pieces of information) are successfully retrieved. It focuses on not missing important results. Higher recall indicates that fewer relevant documents are omitted from the retrieved set, contributing to a more complete and accurate answer generation. This metric is calculated using Equation 7.

$$content\ recall = \frac{|\,Contexts\ retrieved|}{|Total\ of\ reference\ contexts|} \tag{7}$$

4-2- Setup for Generation and Evaluation Model

To ensure that performance differences are attributed solely to the chunking method and not to variations in model capabilities, different embedding models and LLMs are used for generation and evaluation. For generation, documents are then embedded into vector representations using the “BAAI/bge-m3” embedding model [33]. ChromaDB is subsequently employed as the vector store to enable efficient retrieval, and LLaMA 3 70B language model is utilized for answer generation. Besides, a custom system prompt is crafted to ensure that the model generates context-grounded academic responses strictly based on the retrieved documents.

For evaluation, the OpenAI text-embedding-3-small model is utilized to compute semantic similarity, ensuring consistency in the evaluation environment. Additionally, the GPT-3.5 Turbo model is used to perform judgment-based scoring of factual consistency and relevance. A summary of both experiment setups is summarized in Table 2.

Table 2. Summary of Setup of Generation and Evaluation model.

	Generation	Evaluation
Embedding model	BAAI/bge-large-en-v1.5	text-embedding-3-small (OpenAI)
Large Language Model	LLaMA 3 70B (Meta AI)	GPT-3.5 Turbo (OpenAI)
Vector Store	ChromaDB	-

A potential concern with this approach is that using different embedding models for generation and evaluation may introduce inconsistencies, since different models may apply different similarity mechanisms such as cosine similarity, dot product, or distance-based metrics to compute semantic relevance. These differences could theoretically lead to slight variations in how alignment between retrieved and generated content is assessed, particularly affecting metrics like Content Relevance and Answer Relevancy in RAGAS, which rely on embedding similarity comparisons.

However, this effect is minimal for high-quality, modern embeddings, which preserve semantic consistency across models. Using different models for generation and evaluation helps reduce bias, especially in scenarios where shared embeddings may inflate perceived relevance due to self-alignment. Prior studies have shown that retrieval behaviors can vary across embedding models, reinforcing the importance of cross-model setups for fair and unbiased evaluation [34].

To further mitigate the impact of embedding divergence, our evaluation also incorporates LLM-based scoring (i.e., Faithfulness and Answer Accuracy), which offers a complementary and model-independent perspective. These scores are computed through GPT-3.5 Turbo's judgment rather than vector similarity and are thus considered closer to human evaluation. When taken together, this multi-perspective evaluation strategy combining embedding-based and LLM-based metrics provides a more balanced and unbiased assessment of system performance.

To support a fair and interpretable comparison, a set of 50 manually curated questions is designed to reflect typical academic queries encountered in the dataset, as shown in Appendix I. Each question is paired with a human-annotated ground truth answer to serve as the reference for evaluation.

4-3- Parsing-Chunking Baseline Experiment

Unlike prior studies, our method explicitly preserves the logical grouping of semantically coherent content by integrating a customized chunking approach. To assess the contribution of the proposed Structure-Aware Chunking framework, four different baseline configurations are evaluated. Each baseline used the same retriever and generator components but differed in parsing methods and employed the standard Recursive chunking strategy from LangChain [35], as summarized in Table 3. These parsing and chunking strategies represent the most adopted parser–chunking combinations in both open-source and academic RAG systems.

Baseline 1 uses PyPDFLoader, a widely used configuration for unstructured documents, serving as a naive benchmark. Baseline 2 introduces LlamaParse [14] to isolate the contribution of high-fidelity parsing without structural chunking. In this baseline, LlamaParse is explicitly configured with output type as "markdown", which is specifically designed to handle complex table structures with merged headers and nested cells. Baseline 3 replaces the parser with Camelot to evaluate whether enhanced table extraction alone improves performance. Proposed method integrates both accurate table parsing and structure-aware chunking to demonstrate the synergy required for handling semantically dense academic tables. This set of baselines enables a fair and controlled comparison of how different combinations of parsers and chunking strategies impact the quality of retrieved context and generated responses in RAG systems.

Table 3. Baseline Comparison

Baseline	Parser	Chunking
1	PyPDFLoader	Recursive (LangChain)
2	LlamaParse (output = “markdown”)	Recursive (LangChain)
3	Camelot	Recursive (LangChain)
Proposed	**Camelot**	**Proposed Structure-aware Chunking**

5- Results and Discussion

5-1- Evaluation Results

The evaluation results are presented in Table 4 and visualized in Figure 8, comparing all four baselines across six RAGAS metrics. Proposed structure-aware chunking framework achieved the highest scores in all metrics except Faithfulness, demonstrating its effectiveness in handling semantically complex tabular data. However, a closer inspection reveals nuanced trade-offs between different approaches.

Table 4. Evaluation Results

Baseline/Metrics	Baseline 1	Baseline 2	Baseline 3	Proposed Method
Faithfulness	0.68	**0.81**	0.40	0.78
Answer Relevancy	0.72	0.77	0.65	**0.81**
Content Precision	0.57	0.62	0.29	**0.92**
Content Recall	**0.82**	0.73	0.36	**0.82**
Content Relevance	0.77	0.83	0.65	**0.93**
Answer Accuracy	0.32	0.43	0.39	**0.73**

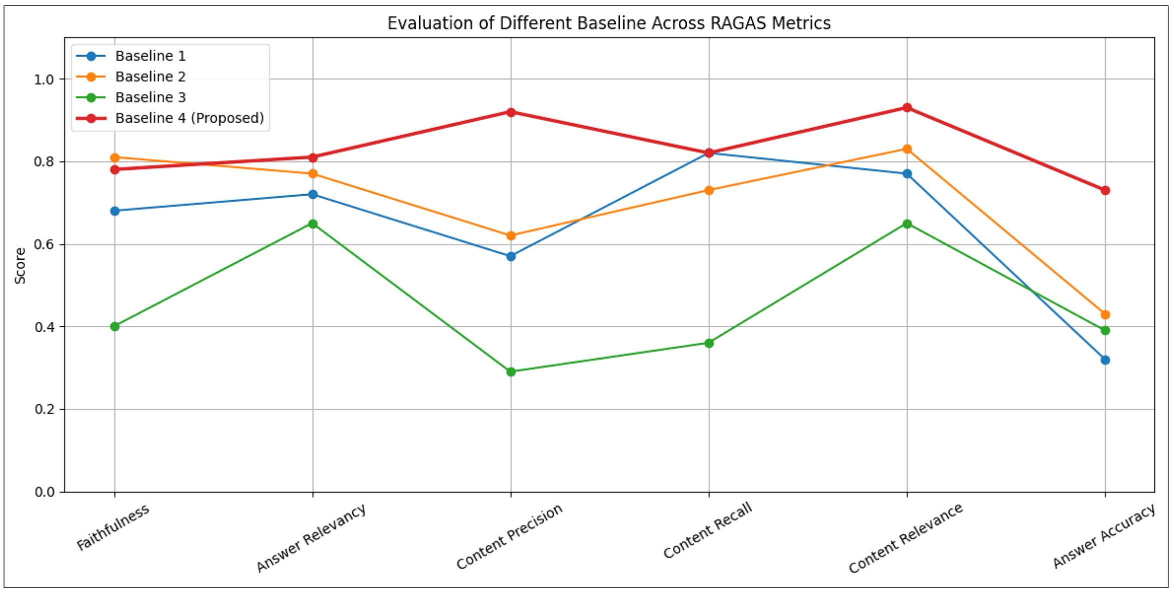


Figure 8. Evaluation of Different Baselines across RAGAS Metrics

Among all methods, Baseline 2 achieved the highest Faithfulness score (0.81). This result highlights the strength of the parser in extracting structured content from complex tables, even when segmented by recursive chunking. This may be due to LlamaParse's ability to preserve intra-cell alignment and generate answers that appear to be supported by the retrieved context. However, this strength can be deceptive. LlamaParse frequently fails to capture critical contextual information in our dataset, which features highly variable table layouts with merged headers and non-repeating labels such as academic term names and category headers. The parser sometimes flattens or misaligns key elements, making it difficult to retain the table's semantic relationships. While the retrieved content may look faithful, it lacks the broader term-level contextual integrity needed for structured academic queries such as "List subjects by term." This structural inconsistency ultimately limits the model's ability to generate responses that are both complete and correctly scoped. It also explains why the Answer Relevancy score for Baseline 2 (0.77) is close to that of our proposed method (0.81). Although both methods retrieved content topically related to the query, only the structure-aware approach ensured that results are correctly grouped by term and enriched with metadata such as credit hours and categories. In contrast, Baseline 2 often returned partially correct but contextually misplaced results, reducing their functional relevance despite local factual alignment.

Despite this, the slightly lower Faithfulness score of the proposed method (0.78) can be explained by the nature of how the metric is computed. Faithfulness is based on the proportion of factually correct statements in the generated response that can be directly supported by the retrieved context. In our framework, the structure-aware chunking strategy enables the system to retrieve a few highly specific and relevant chunks. While this improves retrieval precision, it also reduces the amount of retrieved text available for lexical overlap with the generated answer. As a result, some valid statements in the output may not have a direct match in the retrieved context, leading to a slightly lower Faithfulness score. In contrast, Baseline 2 often retrieves a greater number of broader or less targeted chunks. This increases the likelihood that more words or phrases from the generated response are found in the retrieved context, which artificially boosts the Faithfulness metric, even when the retrieved information is only loosely relevant. It is important to highlight that the proposed method achieves the highest Answer Accuracy (0.73) across all models. This demonstrates that despite retrieving fewer chunks, our method produces correct and complete responses. Therefore, while the Faithfulness score is marginally lower, Answer Accuracy provides a more meaningful measure of our framework's effectiveness in handling structure-dependent academic queries.

While PyPDFLoader (Baseline 1) lacks structural awareness, it achieved a Content Recall score (0.82) similar to the proposed method. This can be attributed to PyPDFLoader's ability to extract nearly all visible text from the table, resulting in broad content coverage. For instance, in fact-based questions such as "What is the subject name of ABC001?", Baseline 1 performs reasonably well, as the necessary keywords are typically present within the retrieved chunks. However, it fundamentally lacks the capability to preserve structural relationships within the rows and columns of the table. During parsing, key layout elements such as term labels and their corresponding subject groupings are often misaligned. This misalignment makes it inherently unsuitable for answering structure-dependent queries like "What subjects are offered in Term D?", where understanding the spatial organization of content is essential. Moreover, this limitation is further amplified during the chunking process, which compounds the loss of semantic coherence. Since recursive chunking splits text based purely on token count, it disrupts any residual structure in the parsed output. Consequently, logically related items such as subjects, terms and categories, become scattered across different chunks, thereby reducing retrieval accuracy.

Besides, the proposed framework outperformed all baselines, achieving perfect scores in both Content Precision (0.92) and Content Relevance (0.93). This demonstrates that the retrieved chunks are not only highly relevant but also semantically complete. These results confirm that structure-aware chunking effectively reduces retrieval noise and improves contextual clarity in the generated responses. Furthermore, the proposed method achieved the highest Answer Accuracy score of 0.73, significantly outperforming Baseline 1 (0.32), Baseline 2 (0.43), and Baseline 3 (0.39). The highest Answer Accuracy directly reflects the system's ability to retrieve from a complex table and generate correct and complete answers based on the retrieved content. Although Baseline 2 slightly outperformed the proposed method in Faithfulness (0.81 vs. 0.78), the margin is minimal and does not translate into superior performance across other critical metrics. In fact, this small gain in Faithfulness comes at the cost of structural integrity, as Baseline 2 often returns topically correct but contextually misaligned responses due to flattened term-subject relationships. In contrast, the proposed method ensures accurate grouping, metadata preservation, and semantic alignment, resulting in more reliable and actionable academic answers. These quantitative findings are further supported by the qualitative examples presented in Section 5.2, where we compare actual generated responses from each method. The examples highlight how structure-aware chunking preserves term-based grouping and metadata alignment, enabling the model to deliver more contextually grounded and complete answers, especially for multi-constraint academic queries. Overall, the results provide strong support for the proposed framework as a robust and semantically aware approach to document understanding, capable of handling structurally rich content and enabling more accurate generation in retrieval-augmented applications.

5-2- Chunks Comparison

Table 5 shows a comparative analysis of responses generated by different baseline methods when answering the representative query, “What subjects are offered in Term D?”, with the ground truth consisting of six specific subjects. Entries highlighted in red indicate incorrectly retrieved subjects that do not belong to Term D. The results demonstrate that PyPDFLoader with Recursive chunking (Baseline 1) often retrieves subjects from unrelated terms due to its flattened and misaligned parsing output. LlamaParse (Baseline 2) fails entirely to retrieve any relevant information for Term D because its parsing process collapses all term labels into a single block, losing the necessary term-subject associations. Camelot (Baseline 3) incorrectly concludes that no subjects are offered in Term D even though it is a strong parser for complex tables. This highlights that a powerful parser alone is insufficient without proper downstream chunking and representation strategies. In contrast, the proposed structure-aware method accurately retrieves all six correct subjects by preserving the semantic relationships between terms and their respective course listings.

Table 5. Sample Output Comparison for Representative Query

Question: What subjects are offered in Term D?

Ground Truth: ABC009, ABC010, ABC011, ABC012, ABC013, and FEE001

Baselines	Retrieved Context (Chunk 1)	Response
1	Year A Year B Year C Trimester A Trimester B Trimester C Trimester D Trimester E Trimester F Trimester G Trimester H Trimester I Term A CH Term B CH Term C CH Term D CH Term E CH Term F CH Term G CH Term H CH Term I CH TotalCategory A ABC001 Subject A X ABC005 Subject E XXABC002 Subject BX ABC006 Subject FXX ABC007 Subject G XXABC008 Subject HXX Category B ABC003 Subject C X ABC009 Subject I X X FEE002 Subject BX ABC025 Subject Y X……	According to the course structure, in Term D, the following subjects are offered: 1. ABC003 Subject C (Term A) 2. FEE002 Subject B (Term E) 3. ABC009 Subject I (Term D) 4. ABC025 Subject Y (Term I)
2	| Term | Course Code| Course Title| CH | | ---------- | -------------- | ---------------|-----| | Term A | ABC001 | Subject A | 4 | | | ABC002 | Subject B | 3 | | | ABC003 | Subject C | 3 | | | ABC004 | Subject D | 3 | | | ABC005 | Subject E | 3 | | | ABC006 | Subject F | 3 |	I'm happy to help you with your academic query. However, I couldn't find any information about Term D in the official document. It seems that the document only provides information about Term A and other courses.
3	Term A CH Term B CH Term C CH Term D CH Term E CH Term F CH Term G CH Term H CH Term I CH Total	According to the academic plan, in Term D, there are no subjects offered. The term is empty.
4 (Proposed)	**Term D (Trimester D):** **1. ABC009 Subject I (X CH), Category B** **2. ABC010 Subject J (X CH), Category B** **3. ABC011 Subject K (X CH), Category C** **4. ABC012 Subject L (X CH), Category C** **5. ABC013 Subject M (X CH), Category C** **6. FEE001 Subject A (X CH), Category E**	**In Term D (Trimester D), the following subjects are offered:** **1. ABC009 Subject I (X CH)** **2. ABC010 Subject J (X CH)** **3. ABC011 Subject K (X CH)** **4. ABC012 Subject L (X CH)** **5. ABC013 Subject M (X CH)** **6. FEE001 Subject A (X CH).**

* Red text indicates incorrectly retrieved subjects or term.

To further demonstrate the effect of structural preservation, Table 6 compares the first two chunk outputs generated by traditional recursive chunking and the proposed structure-aware method. The traditional approach splits content purely based on token count, resulting in fragmented and disorganized outputs. For instance, in chunk

1, subjects from multiple trimesters are mixed together, term labels are misplaced or repeated, and course categories are incomplete or scattered. Such outputs weaken semantic coherence and cause errors in downstream retrieval. In contrast, the proposed structure-aware chunking method produces clear and well-organized outputs. Chunk 1 contains only subjects from Trimester A, each with complete metadata such as credit hours and category. Chunk 2 contains only subjects from Trimester B, which are also structured correctly. This example demonstrates how structure-aware chunking preserves contextual accuracy, improving both retrieval precision and generation reliability.

Table 6. Comparison of Chunking Outputs Generated by Recursive and Structure-Aware Chunking Methods

Chunk No.	Traditional Recursive by LangChain	Proposed Structure-Aware Chunking
1	Year A Year B Year C Trimester A Trimester B Trimester C Trimester D Trimester E Trimester F Trimester G Trimester H Trimester I Term A CH Term B CH Term C CH Term D CH Term E CH Term F CH Term G CH Term H CH Term I CH TotalCategory A ABC001 Subject A X ABC005 Subject E XXABC002 Subject BX ABC006 Subject FXX ABC007 Subject G XXABC008 Subject HXX Category B ABC003 Subject C X ABC009 Subject I X X FEE002 Subject BX ABC025 Subject YX ABC014 Subject N X ABC019 Subject S XXABC004 Subject DX ABC010 Subject JX ABC015 Subject OX ABC020 Subject TXX ABC016 Subject P XX ABC017 Subject Q XX Category CABC011 Subject K X ABC018 Subject	**Trimester A:** **1. ABC001 Subject A (3 CH) Category: Category A** **2. ABC002 Subject B (3 CH) Category: Category A** **3. ABC003 Subject C (3 CH) Category: Category B** **4. ABC004 Subject D (3 CH) Category: Category B** **5. University Requirements A (3 CH) Category: Category H**
2	X ABC018 Subject RXABC023 Subject WX X ABC012 Subject LX ABC021 Subject UXABC024 Subject XX X ABC013 Subject M XABC026 Subject ZX X Category D ABC022 Subject VX X Category EFEE001 Subject AX FEE004 Subject DX X FEE003 Subject C X Category F PRO001 Project X PRO002 Project X X Category G BCA001 Subject A X BCA003 Subject C XXBCA002 Subject BXBCA004 Subject DXXX Category H University Requirements A X University Requirements B XX Total Credits X X X X X X X X	**Trimester B:** **1. BCA001 Subject A (3 CH) Category: Category G** **2. BCA002 Subject B (3 CH) Category: Category G**

5-3- Failure Case Analysis

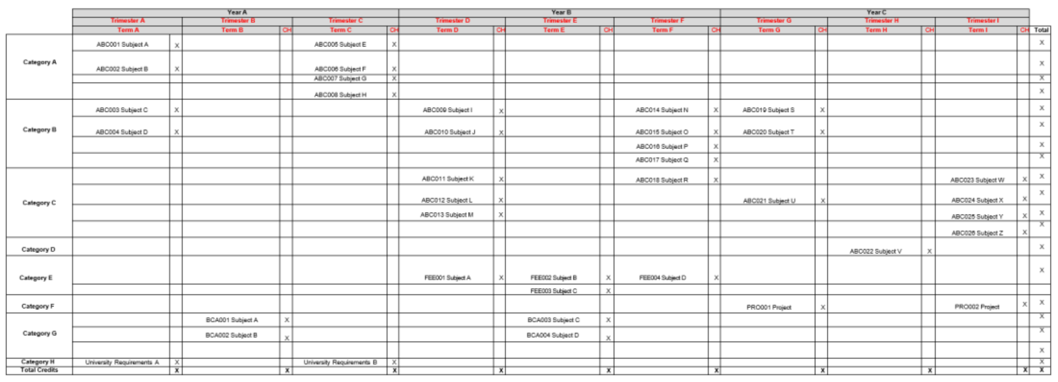
While the structure-aware chunking framework demonstrates improved performance in preserving contextual accuracy and semantic relationships, several specific failure scenarios are observed during internal testing. These limitations arise primarily because the framework is customized to interpret visual alignment as semantic grouping. The spatial arrangement of table cells is used to encode relationships between academic terms, subject codes, credit hours, and categories, as discussed in Table 6.

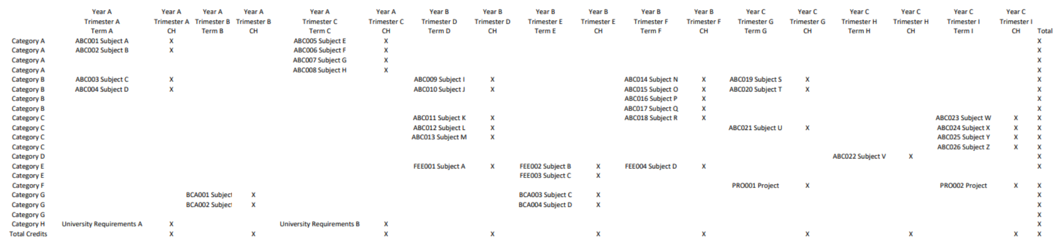
Table 7 summarizes three representative failure patterns identified during the evaluation. These include (a) inconsistent column spans or merged headers, (b) rotated or landscape-oriented pages, and (c) mixed formatting that includes unstructured text. Each of these factors degrades Camelot's parsing performance and disrupts the semantic grouping required for accurate chunk construction. The corresponding output examples for each formatting scenario are illustrated in Figure 9, which presents representative parsing outcomes. Specifically, Figure 9(a) shows a correctly formatted table with consistent grid layout, while Figures 9(b), 9(c), and 9(d) illustrate failure cases caused by inconsistent column spans and merged headers, rotated or landscape-oriented pages, and mixed formatting with unstructured text.

Table 7. Formatting Scenarios and Their Impact on Structure-Aware Chunking

No	Formatting Scenario	Description	Impact on Parsing & Chunking	Chunking Result
(a)	Correctly formatted table	Table uses consistent grid layout, clear cell borders, and standard academic structure.	Camelot lattice accurately extracts structure; chunking produces valid groups.	Success
(b)	Inconsistent column spans or merged headers	Headers are merged across varying numbers of columns or rows inconsistently.	Term or category labels become ambiguous; Subjects are missing or misclassified under the wrong group.	Partially Success
(c)	Rotated or landscape-oriented pages	Tables are rotated or horizontally aligned, disrupting expected row-column structure.	Rows interpreted as columns; subjects and terms misaligned.	Failed
(d)	Mixed formatting with unstructured text	Some documents contain subject listings written as plain text paragraphs rather than structured tables.	Parser cannot extract such content; these subjects are skipped entirely.	Partially Success

(a) Correctly Formatted Table with Consistent Grid Layout




Parsed Output

	Year A Trimester A Term A	Year A Trimester A CH	Year A Trimester B Term B	Year A Trimester B CH	Year A Trimester C Term C	Year A Trimester C CH	Year B Trimester D Term D	Year B Trimester D CH	Year B Trimester E Term E	Year B Trimester E CH	Year B Trimester F Term F	Year B Trimester F CH	Year C Trimester G Term G	Year C Trimester G CH	Year C Trimester H Term H	Year C Trimester H CH	Year C Trimester I Term I	Year C Trimester I CH	Total
Category A	ABC001 Subject A	X			ABC005 Subject E	X													X
Category A	ABC002 Subject B	X			ABC006 Subject F	X													X
Category A					ABC007 Subject G	X													X
Category A					ABC008 Subject H	X													X
Category B	ABC003 Subject C	X					ABC009 Subject I	X			ABC014 Subject N	X	ABC019 Subject S	X					X
Category B	ABC004 Subject D	X					ABC010 Subject J	X			ABC015 Subject O	X	ABC020 Subject T	X					X
Category B											ABC016 Subject P	X							X
Category B											ABC017 Subject Q	X							X
Category C							ABC011 Subject K	X			ABC018 Subject R	X					ABC023 Subject W	X	X
Category C							ABC012 Subject L	X					ABC021 Subject U	X			ABC024 Subject X	X	X
Category C							ABC013 Subject M	X									ABC025 Subject Y	X	X
Category C																	ABC026 Subject Z	X	X
Category D															ABC022 Subject V	X			X
Category E							FEE001 Subject A	X	FEE002 Subject B	X	FEE004 Subject D	X							X
Category E									FEE003 Subject C	X									
Category F													PRO001 Project	X			PRO002 Project	X	X
Category G			BCA001 Subject	X					BCA003 Subject C	X									X
Category G			BCA002 Subject	X					BCA004 Subject D	X									X
Category G																			X
Category H	University Requirements A	X			University Requirements B	X													X
Total Credits		X		X		X		X		X		X		X		X		X	X

Chunk 1:

Trimester A:

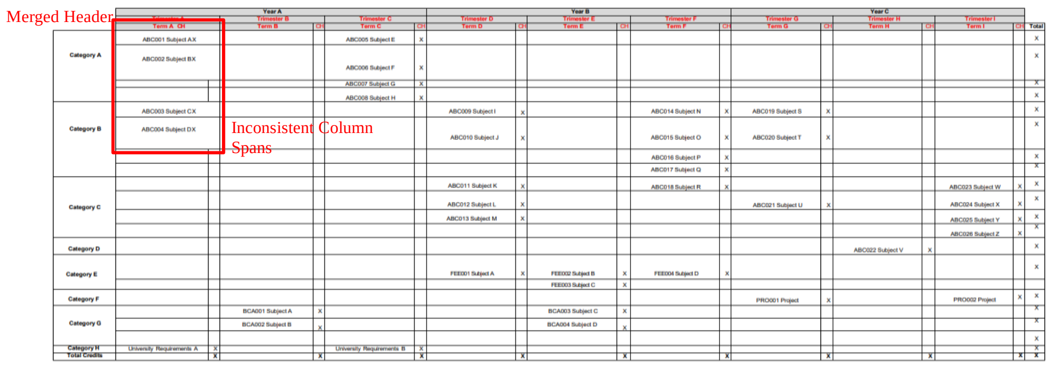
1. ABC001 Subject A (X CH) Category: Category A
2. ABC002 Subject B (X CH) Category: Category A
3. ABC003 Subject C (X CH) Category: Category B
4. ABC004 Subject D (X CH) Category: Category B
5. University Requirements A (X CH) Category: Category H

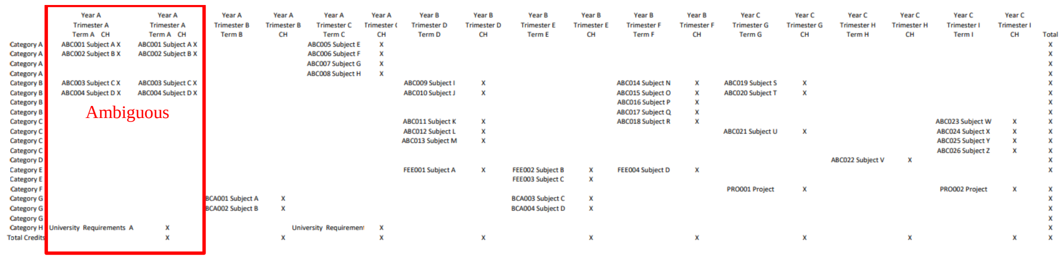
Chunk 2:

Trimester B:

1. BCA001 Subject A (X CH) Category: Category G
2. BCA002 Subject B (X CH) Category: Category G

(b) Failure Case – Inconsistent Column Spans and Merged Headers


Merged Header
Inconsistent Column Spans
Raw Data


Ambiguous
Parsed Output

Chunk Output
4 Subject Missing

Chunk 1:

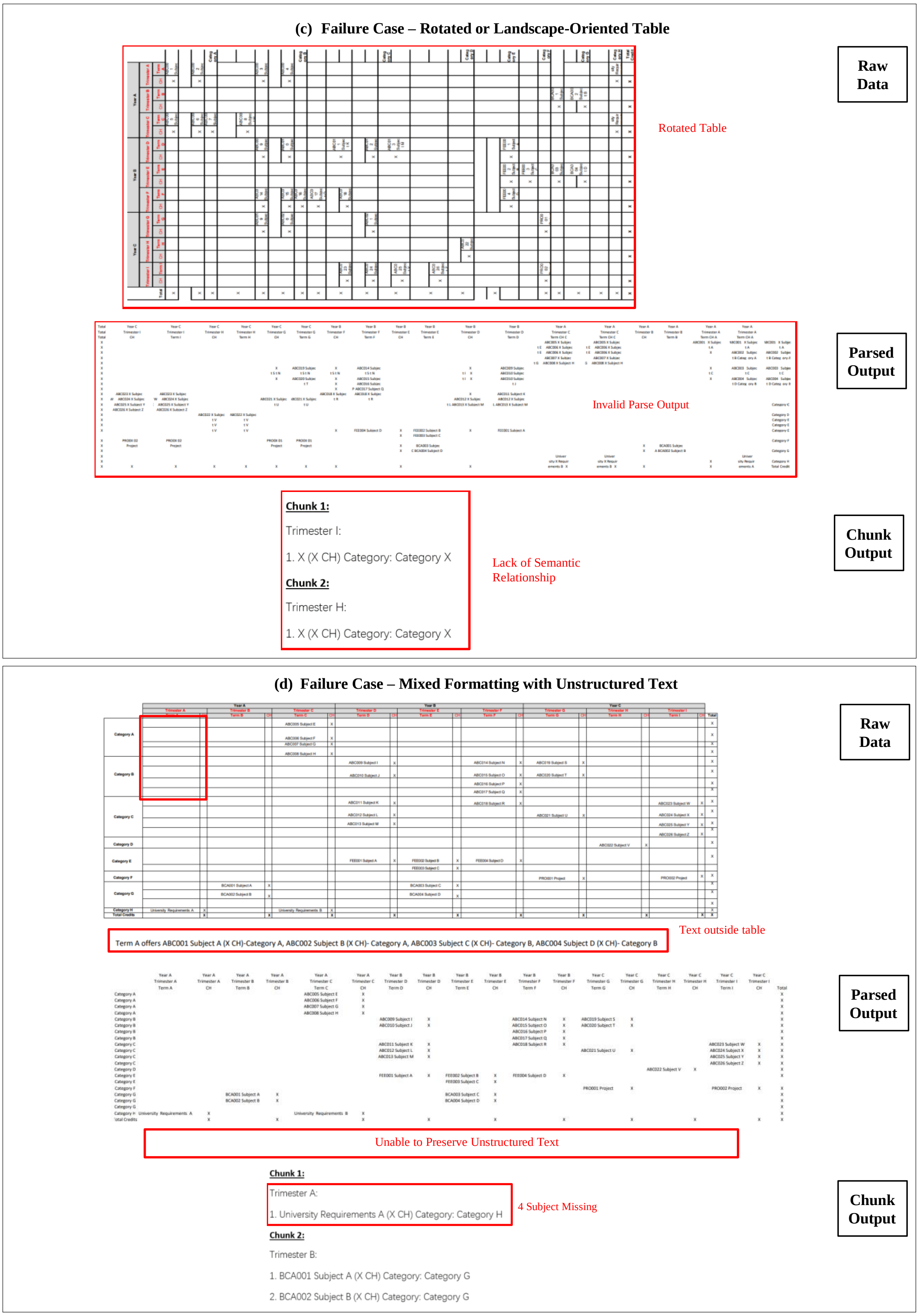
Trimester A:

1. University Requirements A (X CH) Category: Category H

Chunk 2:

Trimester B:

1. BCA001 Subject A (X CH) Category: Category G
2. BCA002 Subject B (X CH) Category: Category G


(c) Failure Case – Rotated or Landscape-Oriented Table
Raw Data
Rotated Table
Parsed Output
Invalid Parse Output
Chunk Output
Lack of Semantic Relationship
(d) Failure Case – Mixed Formatting with Unstructured Text
Raw Data
Text outside table
Parsed Output
Unable to Preserve Unstructured Text
Chunk Output
4 Subject Missing

Chunk 1:

Trimester I:

1. X (X CH) Category: Category X

Chunk 2:

Trimester H:

1. X (X CH) Category: Category X

Term A offers ABC001 Subject A (X CH)-Category A, ABC002 Subject B (X CH)- Category A, ABC003 Subject C (X CH)- Category B, ABC004 Subject D (X CH)- Category B

Chunk 1:

Trimester A:

1. University Requirements A (X CH) Category: Category H

Chunk 2:

Trimester B:

1. BCA001 Subject A (X CH) Category: Category G

2. BCA002 Subject B (X CH) Category: Category G

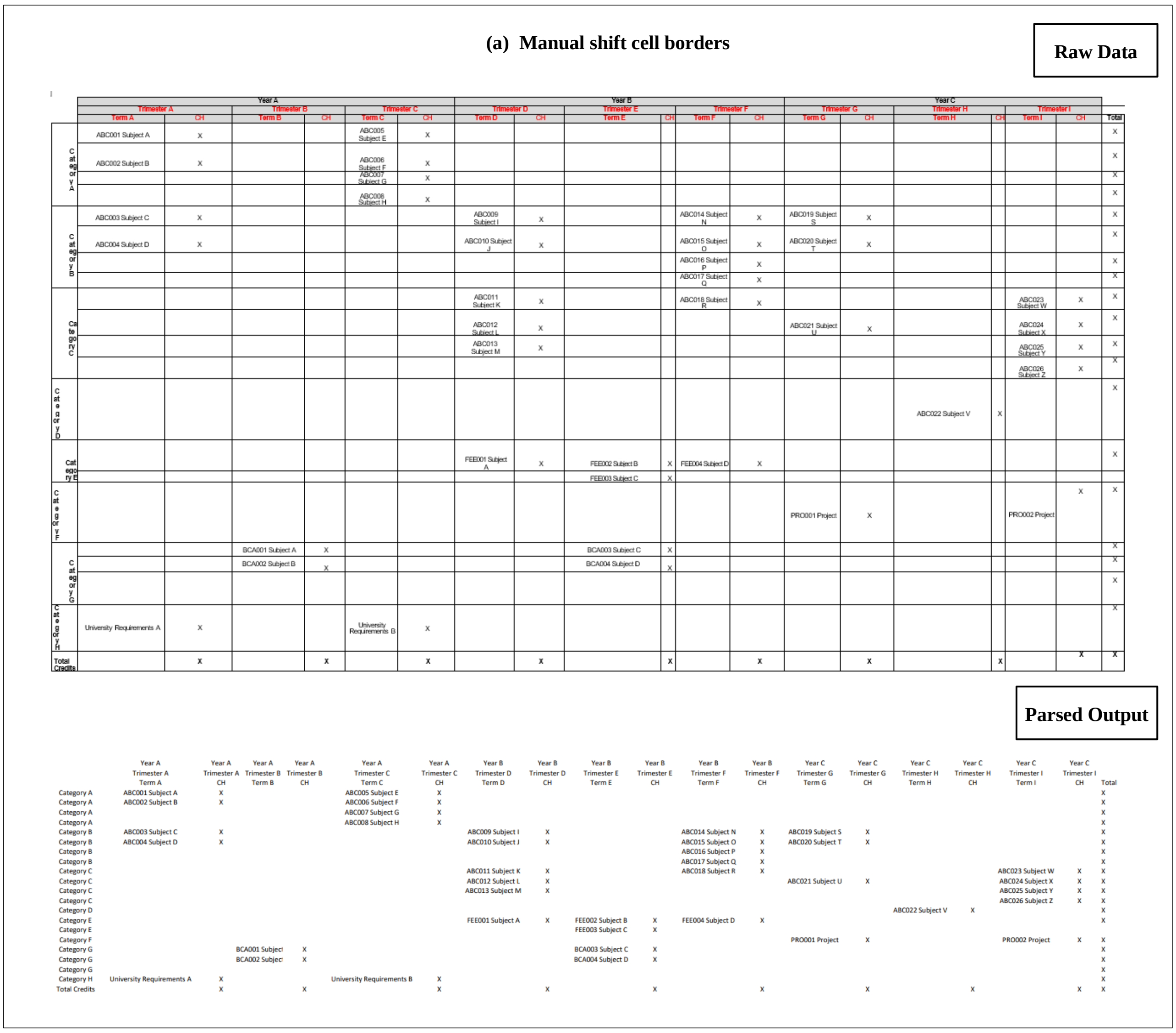
Figure 9. Representative parsing outcomes for different formatting scenarios: (a) Correctly Formatted Table with Consistent Grid Layout; (b) Failure Case – Inconsistent Column Spans and Merged Headers; (c) Failure Case – Rotated or Landscape-Oriented Table; (d) Failure Case – Mixed Formatting with Unstructured Text.

Besides the structural inconsistencies discussed in Table 7, we conducted a focused error case analysis on the system’s sensitivity to minor layout variations, as summarized in Table 8. Although Camelot's lattice mode is precise in extracting structured data from academic tables, it relies on visible cell borders to detect tabular structures. Even subtle disruptions, such as misaligned borders can negatively affect parsing accuracy.

These issues are particularly relevant in our customized framework, which interprets spatial layout as semantic structure. Therefore, minor visual inconsistencies can directly compromise the integrity of the semantic groupings during chunk construction. Figure 10 highlights several examples, demonstrating how subtle variations in table line geometry can cause parsing mistakes or incorrect content grouping.

Table 8. Error Analysis on Sensitivity to Minor Layout Changes in PDF Tables

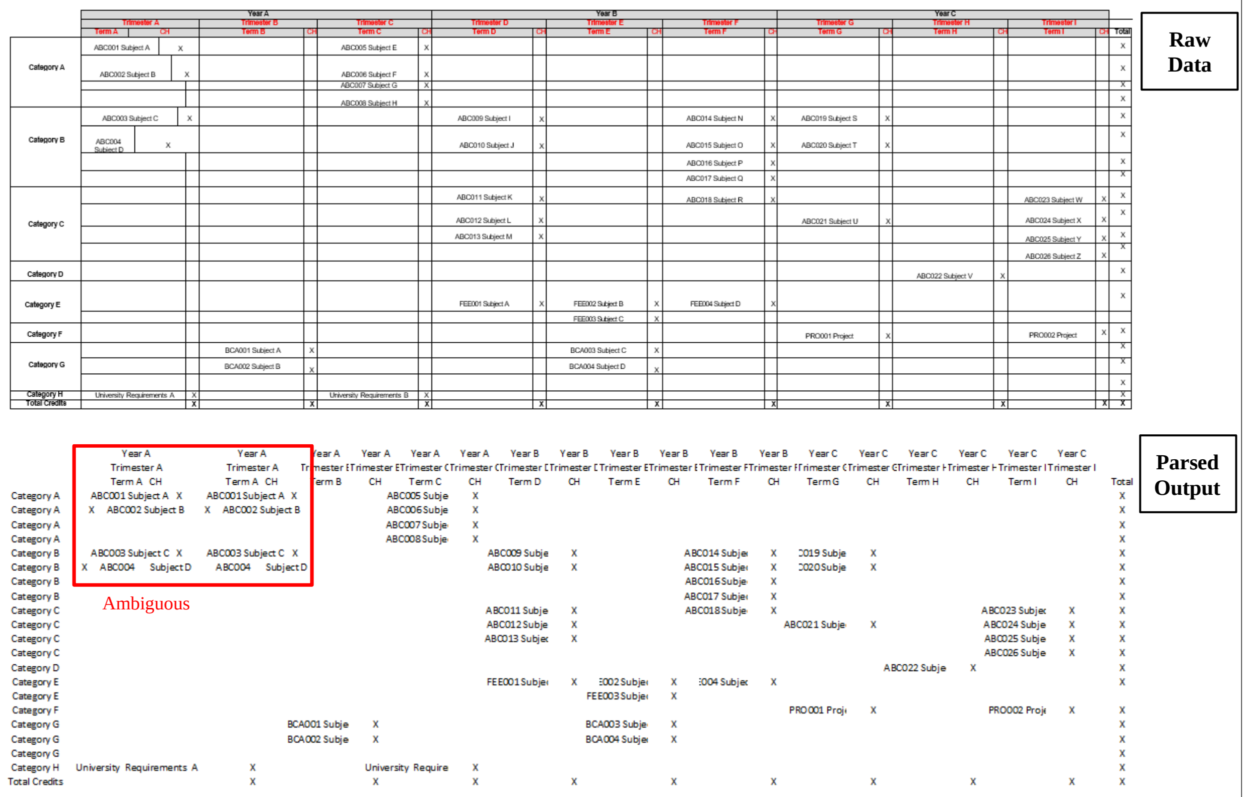
No	Layout Deviation Type	Description	Impact on Camelot Lattice Detection	Parsing Result
(a)	Manual shift cell borders	Cell lines are manually shifted.	Camelot lattice accurately detected and extracted	Success
(b)	Slightly misaligned cell borders	Cell lines are not perfectly aligned.	Camelot fails detect misaligned cells and becomes ambiguous	Failed
(c)	Gaps or breaks in table borders	Vertical and horizontal lines are partially missing or broken, creating discontinuities in cell boundaries	Parser fails to recognize complete cell boundaries, leading to misaligned columns, and merged headers.	Failed



(a) Manual shift cell borders
Raw Data
Parsed Output

	Year A Trimester A Term A	Year A Trimester A CH	Year A Trimester B Term B	Year A Trimester B CH	Year A Trimester C Term C	Year A Trimester C CH	Year B Trimester D Term D	Year B Trimester D CH	Year B Trimester E Term E	Year B Trimester E CH	Year B Trimester F Term F	Year B Trimester F CH	Year C Trimester G Term G	Year C Trimester G CH	Year C Trimester H Term H	Year C Trimester H CH	Year C Trimester I Term I	Year C Trimester I CH	Total
Category A	ABC001 Subject A	X			ABC005 Subject E	X													X
Category A	ABC002 Subject B	X			ABC006 Subject F	X													X
Category A					ABC007 Subject G	X													X
Category A					ABC008 Subject H	X													X
Category B	ABC003 Subject C	X					ABC009 Subject I	X			ABC014 Subject N	X	ABC019 Subject S	X					X
Category B	ABC004 Subject D	X					ABC010 Subject J	X			ABC015 Subject O	X	ABC020 Subject T	X					X
Category B											ABC016 Subject P	X							X
Category B											ABC017 Subject Q	X							X
Category C							ABC011 Subject K	X			ABC018 Subject R	X					ABC023 Subject W	X	X
Category C							ABC012 Subject L	X					ABC021 Subject U	X			ABC024 Subject X	X	X
Category C							ABC013 Subject M	X									ABC025 Subject Y	X	X
Category C																	ABC026 Subject Z	X	X
Category D															ABC022 Subject V	X			X
Category E							FEE001 Subject A	X	FEE002 Subject B	X	FEE004 Subject D	X							X
Category E									FEE003 Subject C	X									
Category F													PRO001 Project	X			PRO002 Project	X	X
Category G			BCA001 Subject	X					BCA003 Subject C	X									X
Category G			BCA002 Subject	X					BCA004 Subject D	X									X
Category G																			X
Category H	University Requirements A	X			University Requirements B	X													X
Total Credits		X		X		X		X		X		X		X		X		X	X

(b) Manual shift cell borders



(c) Gaps or breaks in table borders

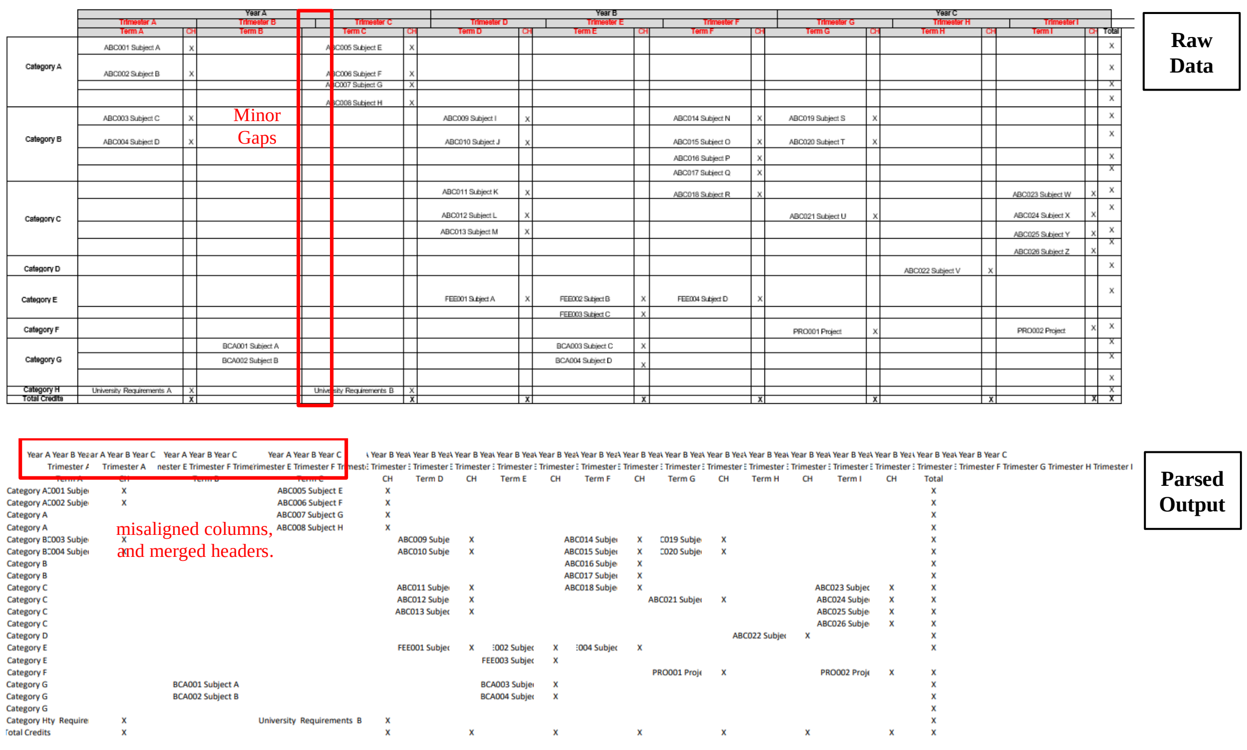


Figure 10. Parsing outcomes under minor layout changes: (a) Manual shift cell borders; (b) Manual shift cell borders; (c) Gaps or breaks in table borders

6- Conclusion and Future Works

This study proposed a customized Structure-Aware Chunking framework for handling complex academic table layouts in Retrieval-Augmented Generation (RAG) systems. Unlike conventional text-based or parser-only methods, the proposed framework combines table parsing via Camelot with Structured-Aware Chunking to group according to terms, categories, and subject–credit hour pairings. Comprehensive evaluation using the RAGAS framework demonstrated that the proposed framework outperformed all alternative baselines across multiple metrics, including Content Precision, Content Relevance, and Answer Accuracy. This study proves that improvements solely in parsing quality are insufficient. Instead, a dedicated chunking process that captures the document's underlying structure is essential for generating accurate and context-relevant responses.

Several limitations have been identified in the current Structure-Aware Chunking framework. The approach relies on Camelot's lattice mode to detect consistent, grid-like table layouts, which may not generalize well to other table formats or layouts found in different types of documents. Additionally, the chunking process relies heavily on the output of Camelot's table parser, in which structural groupings are inferred based on the extracted layout. Consequently, the overall chunking logic is closely linked to the structural integrity of the parsed tables, which limits its effectiveness when applied to documents with unconventional or degraded layouts. In such cases, any deviation in parsing quality can propagate further along the process, necessitating manual reconfiguration of the chunking rules to preserve semantic coherence.

In future work, the Structure-Aware Chunking framework will be extended to handle more diverse and heterogeneous table formats found across various document types. This includes decoupling the chunking logic from fixed layout assumptions and incorporating adaptive parsing mechanisms. Specifically, transformer-based document layout models capable of learning and generalizing spatial and semantic patterns across different table structures will be employed. Additionally, domain-adapted Named Entity Recognition (NER) models will be trained to extract key academic entities such as subject codes, credit hours, term labels, and prerequisite markers independently of specific layout conventions. These enhancements will enable the framework to automatically adapt to different structural formats without requiring manual rule redesign for each new document type.

7- Declarations

7-1- Author Contributions

Conceptualization, X.-K.K. and L.-Y.O.; methodology, X.-K.K.; software, X.-K.K.; validation, X.-K.K., L.-Y.O., and P.-Y.G. formal analysis, X.-K.K., L.-Y.O., and P.-Y.G.; investigation, X.-K.K.; resources, X.-K.K., L.-Y.O., and P.-Y.G.; data curation, X.-K.K.; writing—original draft preparation, X.-K.K.; writing—review and editing, X.-K.K., L.-Y.O., and P.-Y.G.; visualization, X.-K.K.; supervision, L.-Y.O.; project administration, L.-Y.O.; funding acquisition, L.-Y.O. and P.-Y.G. All authors have read and agreed to the published version of the manuscript.

7-2- Data Availability Statement

The data presented in this study are available on request from the corresponding author.

7-3- Funding

This work was supported by the Telekom Malaysia Research and Development under Grant RDTC/221073 (MMUE/230002).

7-4- Institutional Review Board Statement

Not applicable.

7-5- Informed Consent Statement

Not applicable.

7-6- Conflicts of Interest

The authors declare that there is no conflict of interest regarding the publication of this manuscript. In addition, the ethical issues, including plagiarism, informed consent, misconduct, data fabrication and/or falsification, double publication and/or submission, and redundancies have been completely observed by the authors.

8- References

[1] Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., ... & Kiela, D. (2020). Retrieval-augmented generation for knowledge-intensive NLP tasks. Advances in Neural Information Processing Systems, 33, 9459-9474. doi:10.48550/arXiv.2005.11401.

[2] Yepes, A. J., You, Y., Milczek, J., Laverde, S., & Li, R. (2024). Financial report chunking for effective retrieval augmented generation. arXiv preprint, arXiv:2402.05131. doi:10.48550/arXiv.2402.05131.

[3] Shuster, K., Poff, S., Chen, M., Kiela, D., & Weston, J. (2021). Retrieval augmentation reduces hallucination in conversation. In Findings of the Association for Computational Linguistics: Punta Cana, Dominican Republic: Association for Computational Linguistics, EMNLP 2021, 3784–3803. doi:10.18653/v1/2021.findings-emnlp.320.

[4] Chen, X., Wang, L., Wu, W., Tang, Q., & Liu, Y. (2024). Honest AI: Fine-Tuning" Small" Language Models to Say" I Don't Know", and Reducing Hallucination in RAG. arXiv preprint, arXiv:2410.09699. doi:10.48550/arXiv.2410.09699.

[5] Izacard, G., & Grave, E. (2021). Leveraging passage retrieval with generative models for open-domain question answering. In Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume Online: Association for Computational Linguistics, 874–880. doi:10.18653/v1/2021.eacl-main.74.

[6] Kim, K., & Lee, J.-Y. (2024). RE-RAG: Improving open-domain QA performance and interpretability with relevance estimator in retrieval-augmented generation. In Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, Miami, Florida, USA: Association for Computational Linguistics. doi:10.18653/v1/2024.emnlp-main.1236.

[7] Wang, H., Huang, W., Deng, Y., Wang, R., Wang, Z., Wang, Y., ... & Wong, K. F. (2024). Unims-rag: A unified multi-source retrieval-augmented generation for personalized dialogue systems. arXiv preprint, arXiv:2401.13256. doi:10.48550/arXiv.2401.13256.

[8] Manathunga, S. S., & Illangasekara, Y. A. (2023). Retrieval augmented generation and representative vector summarization for large unstructured textual data in medical education. arXiv preprint, arXiv:2308.00479. doi:10.48550/arXiv.2308.00479.

[9] Islam, S. B., Rahman, M. A., Hossain, K. S. M., Hoque, E., Joty, S., & Parvez, M. R. (2024). Open-rag: Enhanced retrieval-augmented reasoning with open-source large language models. arXiv preprint, arXiv:2410.01782. doi:10.48550/arXiv.2410.01782.

[10] Biswal, A., Patel, L., Jha, S., Kamsetty, A., Liu, S., Gonzalez, J. E., ... & Zaharia, M. (2024). Text2sql is not enough: Unifying ai and databases with tag. arXiv preprint, arXiv:2408.14717. doi:10.48550/arXiv.2408.14717.

[11] Roychowdhury, S., Krema, M., Mahammad, A., Moore, B., Mukherjee, A., & Prakashchandra, P. (2024, December). ERATTA: Extreme RAG for enterprise-Table to Answers with Large Language Models. 2024 IEEE International Conference on Big Data (BigData), 4605-4610. doi:10.1109/BigData62323.2024.10825910.

[12] Lin, D. (2024). Revolutionizing retrieval-augmented generation with enhanced PDF structure recognition. arXiv preprint, arXiv:2401.12599. doi:10.48550/arXiv.2401.12599.

[13] Bhat, S. R., Rudat, M., Spiekermann, J., & Flores-Herr, N. (2025). Rethinking Chunk Size for Long-Document Retrieval: A Multi-Dataset Analysis. arXiv preprint, arXiv:2505.21700. doi:10.48550/arXiv.2505.21700.

[14] LlamaParse (2025). AI Document Parsing Software. LlamaIndex, San Francisco, United States. Available online: https://www.llamaindex.ai/llamaparse/ (accessed June 2025).

[15] Liang, Y., Yang, L., Wang, C., Xia, C., Meng, R., Xu, X., Wang, H., Payani, A., & Shu, K. (2025). Benchmarking LLMs for political science: A United Nations perspective. arXiv preprint, arXiv:2502.14122. doi:10.48550/arXiv.2502.14122.

[16] Son, G., Hong, J., Fan, H., Nam, H., Ko, H., Lim, S., Song, J., Choi, J., Paulo, G., Yu, Y., & Biderman, S. (2025). When AI co-scientists fail: SPOT — A benchmark for automated verification of scientific research. arXiv preprint, arXiv:2505.11855. doi:10.48550/arXiv.2505.11855.

[17] Adhikari, N. S., & Agarwal, S. (2024). A comparative study of pdf parsing tools across diverse document categories. arXiv preprint, arXiv:2410.09871. doi:10.48550/arXiv.2410.09871.

[18] Chen, S. A., Miculicich, L., Eisenschlos, J. M., Wang, Z., Wang, Z., Chen, Y., Fujii, Y., Lin, H.-T., Lee, C.-Y., & Pfister, T. (2024). TableRAG: Million-token table understanding with language models. Advances in Neural Information Processing Systems, 37, 74899–74921. doi:10.48550/arXiv.2410.04739.

[19] Gao, Y., Xiong, Y., Gao, X., Jia, K., Pan, J., Bi, Y., ... & Wang, H. (2023). Retrieval-augmented generation for large language models: A survey. arXiv preprint, arXiv:2312.10997. doi:10.48550/arXiv.2312.10997.

[20] Smock, B., Pesala, R., & Abraham, R. (2022). PubTables-1M: Towards comprehensive table extraction from unstructured documents. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 4634-4642. doi:10.48550/arXiv.2110.00061.

[21] Verma, P. (2025). S2 Chunking: A Hybrid Framework for Document Segmentation through Integrated Spatial and Semantic Analysis. arXiv preprint, arXiv:2501.05485. doi:10.48550/arXiv.2501.05485.

[22] Zhao, J., Ji, Z., Fan, Z., Wang, H., Niu, S., Tang, B., Xiong, F., & Li, Z. (2025). MoC: Mixtures of text chunking learners for retrieval-augmented generation system. arXiv preprint, arXiv:2503.09600. doi:10.48550/arXiv.2503.09600.

[23] Merola, C., & Singh, J. (2025). Reconstructing Context: Evaluating Advanced Chunking Strategies for Retrieval-Augmented Generation. arXiv preprint, arXiv:2504.19754. doi:10.48550/arXiv.2504.19754.

[24] Qu, R., Tu, R., & Bao, F. S. (2025). Is semantic chunking worth the computational cost? In Findings of the Association for Computational Linguistics: NAACL 2025, Association for Computational Linguistics, Albuquerque, New Mexico. doi:10.48550/arXiv.2410.13070.

[25] Zhao, B., Ji, C., Zhang, Y., He, W., Wang, Y., Wang, Q., Feng, R., & Zhang, X. (2023). Large language models are complex table parsers. In Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing. Association for Computational Linguistics, 14786–14802. doi:10.18653/v1/2023.emnlp-main.914.

[26] Cremaschi, M., Spahiu, B., Palmonari, M., & Jimenez-Ruiz, E. (2024). Survey on Semantic Interpretation of Tabular Data: Challenges and Directions. arXiv preprint, arXiv:2411.11891. doi:10.48550/arXiv.2411.11891.

[27] Xiao, B., Simsek, M., Kantarci, B., & Alkheir, A. A. (2022). Table structure recognition with conditional attention. arXiv preprint, arXiv:2203.03819. doi:10.48550/arXiv.2203.03819

[28] Anand, R., Paik, H. Y., & Wang, C. (2019). Integrating and querying similar tables from PDF documents using deep learning. arXiv preprint, arXiv:1901.04672. doi:10.48550/arXiv.1901.04672.

[29] Nassar, A., Livathinos, N., Lysak, M., & Staar, P. (2022). TableFormer: Table structure understanding with transformers. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 4614–4623. doi:10.48550/arXiv.2203.01017

[30] Qiu, J., Xiao, F., Wang, Y., Mao, Y., Chen, Y., Juan, X., ... & Wang, M. (2025). On path to multimodal historical reasoning: Histbench and histagent. arXiv preprint, arXiv:2505.20246. doi:10.48550/arXiv.2505.20246.

[31] Madrid-García, A., Benavent, D., Plasencia-Rodríguez, C., Rosales-Rosado, Z., Merino-Barbancho, B., & Freites-Núnez, D. (2025). Optimizing the Clinical Application of Rheumatology Guidelines Using Large Language Models: A Retrieval-Augmented Generation Framework Integrating EULAR and ACR Recommendations. Medrxiv, 2025-04. doi:10.1101/2025.04.10.25325588

[32] Es, S., James, J., Espinosa Anke, L., & Schockaert, S. (2024). RAGAs: Automated evaluation of retrieval augmented generation. In Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics: System Demonstrations. Association for Computational Linguistics, 150–158. doi:10.48550/arXiv.2309.15217.

[33] Chen, J., Zhang, T., Wang, M., Li, Y., Liu, H., Xu, K., & Huang, X. (2024). BGE M3-embedding: Multi-lingual, multi-functionality, multi-granularity text embeddings through self-knowledge distillation. arXiv preprint, arXiv:2402.03216. doi:10.48550/arXiv.2402.03216.

[34] Cao, H. (2025). Writing style matters: An examination of bias and fairness in information retrieval systems. In Proceedings of the Eighteenth ACM International Conference on Web Search and Data Mining (WSDM 2025), 336–344. doi:10.1145/3701551.3703514

[35] Chase, H. (2025). LangChain Software: MIT License. Available online: https://www.langchain.com/ (accessed on January 2026).

Appendix I

Evaluation Questions and Corresponding Ground Truth Answers:

Table A1. Evaluation Questions and Corresponding Ground Truth Answers

No.	Question	Ground Truth
1	What subjects are listed under Term A?	ABC001 Subject A; ABC002 Subject B; ABC003 Subject C; ABC004 Subject D; University Requirements A
2	What subjects are listed under Term B?	BCA001 Subject A; BCA002 Subject B
3	What subjects are listed under Term C?	ABC005 Subject E; ABC006 Subject F; ABC007 Subject G; ABC008 Subject H; University Requirements B
4	What subjects are listed under Term D?	ABC009 Subject I; ABC010 Subject J; ABC011 Subject K; ABC012 Subject L; ABC013 Subject M; FEE001 Subject A
5	What subjects are listed under Term E?	FEE002 Subject B; FEE003 Subject C; BCA003 Subject C; BCA004 Subject D
6	What subjects are listed under Term F?	ABC014 Subject N; ABC015 Subject O; ABC016 Subject P; ABC017 Subject Q; ABC018 Subject R; FEE004 Subject D
7	What subjects are listed under Term G?	ABC019 Subject S; ABC020 Subject T; ABC021 Subject U; PRO001 Project
8	What subjects are listed under Term H?	ABC022 Subject V
9	What subjects are listed under Term I?	ABC023 Subject W; ABC024 Subject X; ABC025 Subject Y; ABC026 Subject Z; PRO002 Project
10	Which Trimester offer Subject R?	Trimester F
11	Which term includes ABC004?	Trimester A
12	Which term includes University Requirements A?	Trimester A
13	Which term offer BCA002?	Trimester B
14	Which term includes Subject H?	Trimester C
15	Which term offer Project?	Trimester G: PRO001 Project; Trimester I: PRO002 Project
16	Which subjects are categorized as Category E?	FEE001 Subject A; FEE002 Subject B; FEE003 Subject C; FEE004 Subject D
17	Which subjects fall under Category G?	BCA001 Subject A; BCA002 Subject B; BCA003 Subject C; BCA004 Subject D
18	In which term is Subject K offered?	Trimester D
19	List all subjects in category B	ABC003 Subject C; ABC004 Subject D; ABC009 Subject I; ABC010 Subject J; ABC014 Subject N; ABC015 Subject O; ABC016 Subject P; ABC017 Subject Q; ABC019 Subject S; ABC020 Subject T
20	List all subjects in category H	University Requirements A; University Requirements B
21	When will offer Subject V?	Trimester H
22	When will offer Subject W?	Trimester I
23	When will offer Subject D?	Trimester A
24	When will offer Subject O?	Trimester F
25 26	When will offer PRO001?	Trimester G
27	When will offer PRO002?	Trimester I
28	When will offer FEE001?	Trimester D
29 30	What subject offer in Term Z?	Trimester Z is not a valid term
31	List all terms available in the course document.	Trimester A; Trimester B; Trimester C; Trimester D; Trimester E; Trimester F; Trimester G; Trimester H; Trimester I
32	When will offer ABC014 Subject N?	Trimester F
33 34	List all subjects under Category A.	ABC001 Subject A; ABC002 Subject B; ABC005 Subject E; ABC006 Subject F; ABC007 Subject G; ABC008 Subject H
35	List all subjects under Category C.	ABC011 Subject K; ABC012 Subject L; ABC013 Subject M; ABC018 Subject M; ABC021 Subject U; ABC023 Subject W; ABC024 Subject X; ABC025 Subject Y; ABC026 Subject Z
36	Which trimester offers ABC009 Subject I?	Trimester D
37	Which trimester offers ABC021 Subject U?	Trimester G
38	List all subjects in Term D.	ABC009 Subject I; ABC010 Subject J; ABC011 Subject K; ABC012 Subject L; ABC013 Subject M; FEE001 Subject A
39	Which trimester includes BCA003?	Trimester E
40	Which trimester offers FEE004 Subject D?	Trimester F
41	List all subjects in Category G.	BCA001 Subject A; BCA002 Subject B; BCA003 Subject C; BCA004 Subject D

42	In which trimester is ABC020 Subject T offered?	Trimester G
43	Which trimester has the most Category B subjects?	Trimester F
44	List of all subjects offered in Trimester I.	ABC023 Subject W; ABC024 Subject X; ABC025 Subject Y; ABC026 Subject Z; PRO002 Project
45	What is the subject name of ABC001?	ABC001
46	Which trimester offers the final project?	Trimester G and Trimester I
47	Which category includes FEE003 Subject C?	Category E
48	List all Category D subjects.	ABC022 Subject V
49	Which trimester includes the subject with code ABC028?	Trimester I
50	Which trimester offers the subject ABC007?	Trimester C