Vrije Universiteit Amsterdam

Universiteit van Amsterdam

Master Thesis

Investigating the Application of Small Language Models for Educational Data Storytelling

Author: Junming Ye (2803441)

1st supervisor:	Dr. A.S.Z. (Adam) Belloum
daily supervisor:	Dr. A.S.Z. (Adam) Belloum
2nd reader:	Ana Oprescu

A thesis submitted in fulfillment of the requirements for the joint UvA-VU Master of Science degree in Computer Science

August 11, 2025

"Esto quod audes"

Abstract

Data storytelling can improve student engagement, but is often limited by the high cost of large language models. Small language models offer a cheaper option, yet their ability to create accurate and engaging educational content is unclear. This thesis studies whether SLMs can generate story-based learning materials from existing curriculum content. We designed a modular retrieval-augmented generation system with two stages: knowledge extraction and story generation. Seven Retrieval-Augmented Generation-based SLMs and 49 model combinations were tested on a custom dataset. Results show that SLMs can create high-quality educational materials when tasks are divided by model strengths. Knowledge extraction models like Phi4-14B ensured factual accuracy, while models like Qwen2.5-7B performed better in narrative generation. Pipelines using different models for each stage outperformed single-model systems. These findings confirm that, with thoughtful design, SLMs are a practical and accessible tool for educational data storytelling.

Contents

List of Figures

List of Tables

1

Introduction

In today's era of big data, the ability to turn raw information into clear and meaningful stories has become a key skill in many fields. This practice is called data storytelling (1). It goes beyond static charts and reports by linking data points into a structured narrative. Such a narrative provides context, explains meaning, and captures the audience's attention. In education, data storytelling has great potential (2). It can simplify complex ideas, make abstract concepts easier to understand, and help students learn more effectively. By presenting knowledge as a story, teachers can improve both knowledge retention and classroom engagement.

While LLMs dominate AI-driven content generation, their computational cost limits educational applications—a gap where Small Language Models (SLMs) may offer a practical alternative. However, LLMs also have clear drawbacks: they are very large, require heavy computation, and cost a lot to run. These problems make them difficult to use in many schools and universities (3). To overcome these limitations, researchers have started to focus on Small Language Models (SLMs). These models have fewer than 15 billion parameters and are much lighter and cheaper to run. They can still generate high-quality text while using far fewer resources.

Although LLMs have been widely studied, there is little research on using SLMs for educational data storytelling. This task demands three key elements: factual precision, logical coherence, and pedagogical intuition. The goal of this thesis is to explore whether SLMs can be used to create structured, story-based learning materials from existing educational content.

This study investigates the problem in a systematic way, guided by the following research questions.

1.1 Research Questions

This study addresses the following research questions:

1. **RQ1:** To what extent can Small Language Models be used to generate coherent and useful educational data stories from structured materials?
2. **RQ2:** How do different Small Language Models, and their combinations in a multi-step pipeline, compare in performance, factual accuracy, and narrative quality?

RQ1 examines whether SLMs can follow instructions, stay factually correct, and produce narratives suitable for the classroom. And RQ2 aims to find the strengths and weaknesses of different SLMs and combinations, and to see which setups work best for knowledge extraction and story generation.

1.2 Contributions

To answer these questions, this thesis makes the following contributions:

1. We design and implement a modular end-to-end system for educational data storytelling using Small Language Models. The pipeline uses Retrieval-Augmented Generation (RAG) and a chained-prompting strategy optimized for SLMs.
2. Our system performs a large-scale evaluation of seven SLMs within the system, and shows how the knowledge extraction and story generation stages affect each other and provide clear guidance for choosing the best models.

The source code for this project is available at the GitHub repository[1]

1.3 Thesis Structure

The rest of this thesis is organized as follows. Chapter 2 introduces key concepts such as Retrieval-Augmented Generation, Small Language Models, and Data Storytelling. Chapter 3 reviews related research on Large Language Models in education, prompt engineering, storytelling, and RAG applications, highlighting research gaps. Chapter 4 describes the design of the RAG-based data storytelling system, including knowledge extraction and story generation modules. Chapter 5 presents evaluation results comparing different Small

[1] `https://github.com/yip-jm/eduDS`

Language Models and their combinations. Chapter 6 discusses the findings, limitations, and future research directions. Chapter 7 concludes by summarizing the contributions and key insights.

2

Background

2.1 Retrieval-Augmented Generation

Retrieval-Augmented Generation (RAG) is a technique that combines information retrieval with generative models (4), enabling large language models (LLMs) to incorporate authoritative external knowledge before generating responses. By leveraging this external information, RAG significantly improves the relevance, accuracy, and utility of generated content—especially in specialized or knowledge-intensive tasks—while effectively mitigating hallucination issues without requiring model retraining (5, 6). This approach not only maximizes the potential of LLMs but also broadens their applicability across a wide range of professional domains.

2.1.1 Architectures and Developments

The RAG architecture primarily consists of three key components: indexing, retrieval, and generation. In the indexing stage, documents are divided into chunks, which are then transformed into vectors using an embedding model and stored in a vector database. During retrieval, the user's query is compared against the vector database to find the top-K most relevant chunks based on similarity. In the generation stage, the retrieved content is combined with the query and input into LLMs to produce the final answer.

According to its development, Gao et al. categorize RAG into three stages: Naive RAG, Advanced RAG, and Modular RAG(6). Naive RAG follows the traditional pipeline, consisting of indexing, retrieval, and generation. Advanced RAG enhances the retrieval process by introducing pre-retrieval and post-retrieval process(7, 8, 9), aiming to improve retrieval efficiency and increase the utilization of retrieved chunks. Modular RAG decouples components such as retrieval and generation, with each module responsible for a specific

function. In the Modular RAG designed by Gao et al., an orchestration mechanism is introduced to dynamically select the subsequent steps in the RAG workflow(10).

2.1.2 Optimization

The optimization of RAG can be divided into three key stages based on indexing, retrieval, and generation, specifically involving input enhancement, retriever enhancement, and generator enhancement.

Input enhancement primarily includes two methods: query transformation and data augmentation. Query transformation involves generating pseudo-documents based on the original query and using them as new inputs for retrieval, thereby improving retrieval accuracy (8). Data augmentation preprocesses the data before retrieval, where structured representations enhance the precision and usability of the information (11).

For retriever enhancement, multiple approaches can be employed to improve the quality of retrieved content and to better utilize in-context learning capabilities, including Recursive Retrieval, Chunk Optimization, Finetuning the Retriever, Hybrid Retrieval, and Reranking. Recursive retrieval performs multiple rounds of search to obtain richer and higher-quality content. Chain-of-Thought (CoT) reasoning in ReACT enables multi-step retrieval through query decomposition, allowing access to more comprehensive information (12). Chunk optimization techniques improve retrieval results by adjusting the size of text chunks. Sliding window is an effective method that captures contexts to enhance retrieval relevance, while LlamaIndex organizes documents into a tree structure and performs hierarchical retrieval, automatically merging results layer by layer (13). As the core component of a RAG system, the retriever can further enhance its representation capabilities in specialized domains by using high-quality embedding models such as BGE and AngIE (14, 15). Hybrid retrieval integrates sparse and dense retrievers to handle diverse query types better, while reranking the retrieved results increases the visibility and utility of critical information (16).

In the generation stage, prompt engineering can guide large language models to produce more relevant and higher-quality outputs. Specifically, this includes zero-shot, few-shot, role prompt, instruction prompt, and chain-of-thought (CoT) prompting methods. Zero-shot prompting refers to using a prompt without providing any examples, allowing the large language model to perform the task directly (17). This approach is also considered one of the key settings for evaluating the inherent capabilities of large language models. In Few-shot prompting, few-shot examples provide large language models with in-context learning, guiding them in both the format and content of the desired output (18). Role prompting

assigns the model a specific role, which helps enhance its performance on targeted tasks by aligning its responses with the expectations of that role (19). Compared to a basic prompt, instruction prompting provides the model with a comprehensive description of the task, enabling it to generate outputs that are more precise and relevant (20). CoT prompting encourages the model to generate intermediate reasoning steps or a "chain of thought," simulating the human problem-solving process to enhance its ability to perform complex reasoning tasks (21).

2.1.3 Evaluation

For the evaluation of the RAG system, it can be primarily divided into two aspects: retrieval, generation. The retrieval component is assessed based on how effectively it identifies and ranks relevant documents from a large corpus, which is crucial for grounding the generated content in factual evidence. Metrics such as Recall, Effective Information Rate (EIR), and normalized discounted cumulative gain (nDCG) are commonly used to evaluate retrieval quality. On the other hand, the generation component is evaluated by examining the quality, relevance, coherence, and factual consistency of the final output(22, 23).

2.2 Small Language Models

Large Language Models (LLMs) have demonstrated impressive performance across a wide range of tasks. However, their massive parameter sizes and high computational demands present significant challenges in terms of time and computational cost. To address these challenges, Small Language Models (SLMs), which typically have fewer than one billion parameters, have emerged as lightweight alternatives. With lower computational overhead and strong practicality, SLMs are becoming an increasingly important choice for real-world applications.

The development of SLMs relies on a variety of technical methods to reduce model size while maintaining strong performance. Knowledge distillation allows smaller models to learn from the outputs or internal representations of larger models, enabling effective domain-specific enhancement (24). Weight pruning reduces computational overhead by removing redundant parameters without significantly sacrificing accuracy (25). Quantization uses low-precision numerical formats to lower memory consumption (26). Efficient network architectures such as DistilBERT (27) and MobileBERT (28) achieve reduced model size and improved computational efficiency through techniques like parameter sharing and bottleneck layers, while preserving model depth.

As shown in Table 2.1, in the open-source community, several well-optimized models have emerged, including DeepSeek-LLM, Gemma, Qwen2.5, OpenChat, Llama3.1, OLMo2, and Phi-4. DeepSeek-LLM is known for its strong capabilities in code generation and mathematical reasoning, making it well-suited for complex tasks (29). Gemma performs exceptionally well in academic benchmark tests for language understanding, reasoning, and safety (30). Llama3.1 enhances long-context retention and multilingual generalization while aligning better with safety benchmarks (31). Olmo2 is a fully open and transparent research model designed for reproducible scientific evaluation (32). Openchat is a conversational AI model designed to enhance open-source language models using mixed-quality data, with optimizations for instruction following and multi-turn dialogue (33). Qwen2.5, trained on high-quality bilingual data with an advanced tokenizer, excels in multilingual and instruction-following tasks (34). and Phi-4, developed by Microsoft, achieves a balance between performance and efficiency through architectural optimizations and high-quality training data, demonstrating strong capabilities in reasoning and mathematics (35).

SLMs	Deepseek-llm	Gemma	Llama3.1	Olmo2	Openchat	Qwen2.5	Phi-4
Size	7B	7B	8B	7B	7B	7B	14B
Provider	Deepseek	Google	Meta	Allenai	Tsinghua	Alibaba	Microsoft

Table 2.1: Basic information of several open-source SLMs

2.3 Data Storytelling

In the era of abundant data, the ability to extract meaningful narratives from complex datasets has become a crucial interdisciplinary skill. Traditional methods of data presentation, which focus on static visualizations and numerical reports, have now evolved into a more dynamic and context-sensitive practice known as data storytelling. This approach integrates data analysis, narrative construction, and visual presentation, which not only aids understanding but also enhances engagement, persuasion, and decision-making.

Based on my previous research (36), data storytelling is typically conceptualized as a three-stage process comprising data exploration, story construction, and story narration. The exploration phase involves descriptive, diagnostic, predictive, and prescriptive analytics, enhanced by exploratory data analysis. Story construction entails two key components: narrative modeling and visual representation. A variety of narrative frameworks, such as Freytag's Pyramid, Aristotle's three-act structure, and Campbell's Hero's Journey, are used to shape coherent, emotionally resonant stories. Visual representation must balance

clarity, salience, and audience engagement, often guided by design heuristics rooted in human visual perception. The final stage, narration, focuses on the delivery of the story, ranging from linear, author-driven formats to interactive, reader-driven designs. AI is increasingly integrated into this stage, enabling adaptive storytelling, personalized content, and immersive environments through AR/VR technologies.

Applications of data storytelling are notably domain-specific, with significant methodological and strategic variations observed across journalism, education, finance, and medicine, as shown in Table 2.2. In journalism, data storytelling is used to communicate complex societal issues to the general public, employing structures like the 5W-1H model and integrating interactive visualizations to enhance comprehension and engagement. In education, digital storytelling serves as a pedagogical tool, emphasizing emotional connection and student co-creation through multimedia narratives. In finance, storytelling frameworks are designed to support decision-making, with curated dashboards and structured reports aimed at different stakeholders. In medicine, narrative and visual techniques are applied to communicate clinical data and treatment information effectively to both professionals and patients, often employing 2D/3D interactive tools to enhance accessibility and comprehension.

Applications	**Journalism**	**Education**	**Finance**	**Medicine**
Data Exploration	descriptive	*abstract concepts*	descriptive, diagnostic, predictive	descriptive, diagnostic
Story Construction	linear structure; 5W-1H framework	7 key elements, 4 key steps, student participation	5 key attributes, three-stage policy model	characters, plot, themes, Freitag's Pyramid, Hero's Journey
Visual Representation	annotated charts, interactive graphics, multimedia elements	multimedia integration (images, audio, video)	line charts, dynamic graphs, dashboards	interactive 2D/3D visualizations, medical animations
Story Narration	linear/hybrid structures	first-person perspective, question-driven	problem-solution structure, tailored to audience	patient-centered, reader-driven model, interactive slides

Table 2.2: Comparison of Data Storytelling Methods Across Different Applications

Recent advancements in artificial intelligence have profoundly reshaped the landscape of data storytelling. AI technologies now contribute not only to automated data analysis but also to the generation of narrative content, visualization recommendations, and delivery customization. Despite these innovations, scholars emphasize that human oversight remains indispensable to ensure contextual appropriateness, narrative coherence, and ethical integrity, particularly as AI systems may lack domain-specific understanding and transparency. Therefore, the field is moving toward a paradigm of human–AI collaboration,

where automation enhances, rather than replaces, human interpretive and communicative capacities.

The literature study consistently conceptualizes data storytelling as a three-stage process: data exploration, story construction, and story narration, supported by both narrative theory and visual design principles. While domain-specific adaptations exist, such as the participatory and multimedia-rich approach in education, current research still shows several gaps relevant to this thesis: (1) Most educational applications prioritize narrative engagement but lack systematic frameworks to ensure factual accuracy and structured knowledge delivery, particularly when transforming abstract curriculum concepts into coherent stories. (2) The integration of AI, especially Small Language Models, remains underutilized for end-to-end educational storytelling pipelines. (3) Although human–AI collaboration is widely acknowledged as necessary for contextual appropriateness and ethical integrity, practical, scalable systems embedding retrieval-augmented generation into storytelling for classroom use are scarce. This thesis aims to address these gaps by building a modular RAG-based system that brings together factual accuracy, narrative coherence, and alignment with teaching goals, using SLMs to create accessible and high-quality educational data stories.

3

Related Work

Large Language Models (LLMs) have demonstrated significant potential in various areas, particularly in education. They can create content, provide personalized feedback, and assist in building interactive learning environments. Due to these abilities, numerous studies have been conducted in this field. In this chapter, we review previous research from four main areas that are closely related to our topic: (1) how LLMs are used in education, (2) prompt engineering methods for educational purposes, (3) using storytelling and narratives in teaching, and (4) Retrieval-Augmented Generation (RAG) in smart educational systems. At the end, we point out the research gaps that our study hopes to fill.

3.1 Applications of LLMs in Education

The integration of LLMs into educational settings has enabled novel teaching and learning paradigms. George (37) outlined four key areas where generative AI can benefit postgraduate education: personalized learning, automated feedback, intelligent research support, and content creation. However, the study remains largely exploratory and lacks a concrete system architecture for real-world deployment. In contrast, our work offers a fully modular and deployable framework tailored for classroom integration, focusing on practical usability rather than conceptual potential.

Similarly, Thüs et al. (38) developed a RAG-based system *OwlMentor*, which demonstrates how AI can support metacognitive strategies (such as self-questioning, summarizing, and self-explanation) and improve answer accuracy through semantic retrieval and context generation. In contrast, our work investigates the use of small models for adaptive, instructor-driven teaching. We enable teachers to integrate their own course materials

and leverage data storytelling to interpret students' learning trajectories and conceptual understanding.

3.2 Prompt Engineering Strategies for Educational Tasks

Prompt engineering plays a critical role in adapting LLMs to educational goals. Chen et al. (39) conducted a systematic review of prompt strategies used in K–12 STEM education, categorizing techniques such as zero-shot learning, chain-of-thought prompting, and retrieval-augmented prompting. These methods were evaluated across multiple tasks, including tutoring, curriculum alignment, and concept explanation, with generally positive outcomes.

Building on these findings, our system integrates prompt engineering as a core design element. We develop tailored prompt templates aligned with instructional goals and apply them systematically across stages of content generation, from factual retrieval to narrative synthesis. This integration improves the factuality, fluency, and educational relevance of outputs, even when using smaller, more efficient language models.

3.3 Storytelling and Narrative Generation in Education

Storytelling has long been recognized as an effective pedagogical method to improve student engagement and conceptual understanding. Jiang et al. (40) applied narrative generation techniques in legal education, using LLMs to simplify complex legal concepts for non-experts. Their results showed improved comprehension through narrative framing, although their approach remained confined to a specific domain.

Our work generalizes this idea by introducing a storytelling pipeline adaptable to STEM education. Through the use of guided prompts and model scaffolding, we transform structured knowledge into pedagogically effective teaching stories. Furthermore, we evaluate several small-scale LLMs in terms of narrative fluency and factual accuracy to ensure that the generated content aligns with instructional goals.

In addition, Eldan and Li (41) demonstrated that small LLMs, when guided with carefully designed prompts, are capable of producing coherent and fluent short stories. Inspired by their findings, our system incorporates prompt templates, predefined structural cues, and content constraints to support smaller models in generating instructional narratives that are both imaginative and educationally meaningful.

3.4 RAG in Educational Systems

RAG has emerged as a key mechanism for combining external knowledge retrieval with generative outputs, enhancing both factual accuracy and contextual relevance. Its use in education is growing, particularly for question answering and reading support. Wang et al. (3) provided a comprehensive review of LLMs in education, examining use cases such as intelligent tutoring, automated grading, question generation, and content summarization. Of particular relevance to our work is their identification of RAG as a promising approach to enhance the factual reliability of LLM outputs in educational contexts. While their study highlights the potential of such methods, our work advances this discourse by operationalizing RAG within a structured lesson generation pipeline, with a specific focus on the creation of story-based instructional content that aligns with pedagogical goals.

For instance, the *AI-University* platform (42) leverages LLMs and RAG to align generative content with instructional goals in scientific education, improving alignment with learning standards and domain knowledge. While the system showcases the utility of RAG for educational support, it primarily enhances isolated tasks rather than generating structured content such as lesson plans or stories.

Our work expands the role of RAG from a supportive backend function to a central design component. We use RAG not only to ensure factual accuracy but also to structure content progression and guide the generation of coherent, multi-part teaching narratives.

Furthermore, Pan et al. (43) introduced a method to guide generative storytelling using structured knowledge graphs. Their system improved coherence and topic relevance in generated narratives, while it may be less suitable for introspective, emotion-driven stories, particularly when using smaller LLMs.

3.5 Summary and Research Gaps

Across these thematic areas, prior research highlights the significant potential of LLMs in education. However, several gaps remain:

- **Lack of practical, scalable systems**: Many studies stop at proof-of-concept systems or domain-specific tools, without addressing real-world classroom usability or modularity.
- **Underutilization of storytelling**: Despite its pedagogical value, storytelling is rarely treated as a central design component in educational AI systems.

- **Prompt engineering fragmentation**: Prompt design is often addressed in isolation, without integration into full pipelines for educational content generation.
- **Limited role of RAG**: Although increasingly used for factual enhancement, RAG is seldom embedded as a core mechanism driving structured educational outputs.
- **Over-reliance on large models**: Many systems rely on high-resource LLMs, limiting accessibility and scalability.

Our work addresses these challenges through an end-to-end, modular system that integrates RAG, prompt engineering, and small language models to generate educational stories grounded in curriculum-aligned knowledge. By embedding storytelling and factual retrieval within a guided prompting framework, we offer a novel, lightweight solution for real-world educational deployment.

4

Design

This chapter presents the design of the proposed RAG-based data storytelling system for educational applications. It introduces the overall architecture and describes each major component in detail, including the knowledge extraction module, the story generation and narration module, the dataset construction process, and the evaluation methodology. Additional details, including step-by-step implementation guidelines and recommendations for adapting the system to different contexts, are provided in Section 6.3.

4.1 Top-level Design

The proposed system is designed as a complete pipeline for generating educational data stories based on structured domain knowledge. It combines RAG with modular story generation and interactive narration, aiming to support teachers in preparing lesson content that is both accurate and engaging. The system's architecture, shown in Figure 4.1, is modular and interpretable, consisting of three main components: **Knowledge Extraction**, **Story Generation**, and **Story Narration**.

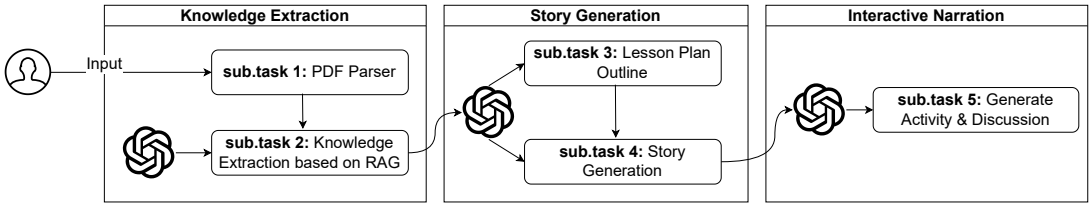

Figure 4.1: The Top-level Design of RAG-based Data Storytelling System

4.2 Knowledge Extraction

The knowledge extraction module is the first and foundational step in the overall system architecture. Its objective is to convert raw learning materials, such as textbooks and PDF documents, into semantically meaningful, structured chunks of information suitable for downstream story generation and interactive narration. This section details the design choices and workflow of the knowledge extraction process, as illustrated in Figure 4.2

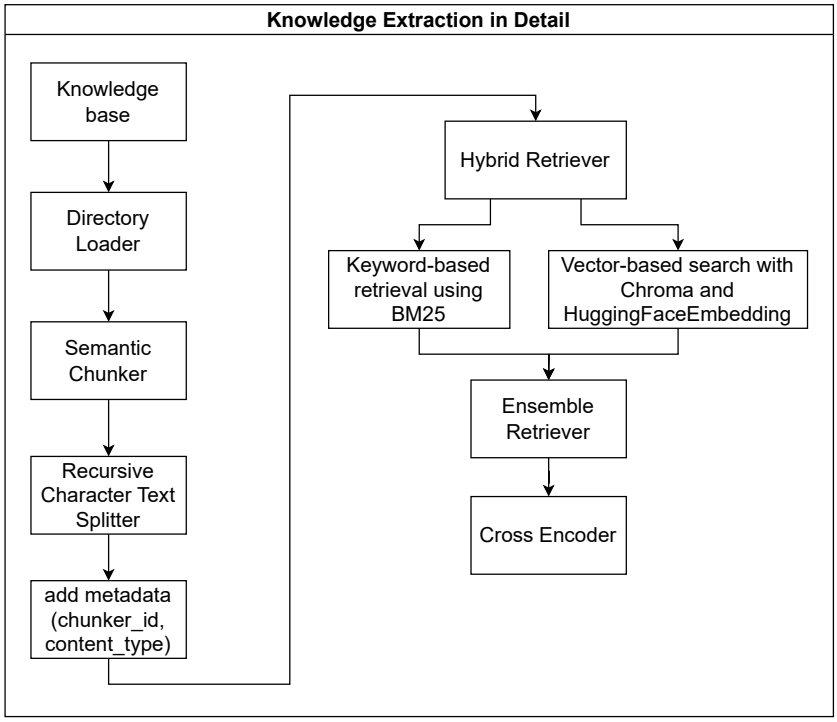

Figure 4.2: Knowledge Extraction in Detail

4.2.1 Document Processing and Chunking

The process begins by collecting input documents into the system's knowledge base. These documents are loaded using a directory loader and processed through a two-stage chunking pipeline, which is designed to maintain both semantic clarity and proper text length.

To deal with the complexity of educational documents—such as multi-column layouts, tables, and figures—we use an open-source tool called minerU for PDF parsing. minerU performs well in preserving the original document structure and ensures clear content extraction.

After parsing, the documents go through a dual-stage segmentation process:

- **Semantic Chunker:** First, the documents are split into semantically meaningful parts using embedding-based similarity. This ensures that each chunk focuses on one main idea or concept.
- **Recursive Character Text Splitter:** Then, a second layer of segmentation is applied. This uses a character-based recursive approach to keep chunks within a length limit, making them suitable for downstream processing.

Each chunk is then tagged with metadata, including a unique *chunker_id* and an inferred *content_type*, which helps in tracking and retrieving relevant information later.

4.2.2 Hybrid Information Retrieval

To support RAG, the system implements a hybrid retrieval strategy that integrates both keyword-based and semantic search mechanisms. This design is motivated by the strengths and limitations of each method:

- **Keyword-based retrieval (BM25):** This method is effective for finding exact matches, especially when the query contains technical terms from the educational domain.
- **Vector-based retrieval (Chroma + HuggingFace Embedding):** This allows the system to retrieve semantically similar content, even when the query uses different words or phrasing.

The results from these two retrieval methods are merged by a Hybrid Retriever, which ranks them together to improve accuracy and coverage.

4.2.3 Re-ranking via Cross Encoder

After retrieval, all candidate chunks are passed into a Cross Encoder for re-ranking. Unlike the individual scoring methods used in sparse or dense retrieval, the Cross Encoder processes the query and chunks together, which improves the accuracy of relevance scoring.

Only the top-k documents with the highest relevance scores are retained for the generation phase. This step ensures that only contextually appropriate and high-quality content is used during narrative construction.

4.3 Story Generation and Narration

The story generation and narration component is responsible for transforming structured domain knowledge into engaging, pedagogically effective educational stories. Given the token limitations of SLMs, our system does not generate the entire story in one prompt. Instead, it adopts a modular pipeline approach using chained prompts as shown in Figure 4.3. This strategy allows complex tasks to be broken down into smaller, more manageable steps, ensuring higher quality and coherence in the generated output. The overall design follows four core principles:

- **Decoupling:** The system first builds a general lesson outline, and teachers can then request detailed story modules as needed.
- **Mapping & Transformation:** Structured knowledge fields are carefully mapped to story elements based on storytelling theory.
- **Interactive Generation:** Teachers are actively involved. They can review, select, and adjust each generated output step by step.
- **Focus & Brevity:** Each prompt is task-specific, keeping the output clean, relevant, and within the token limits of small models.

4.3.1 Story Generation

The system employs a chained, modular prompt strategy: each teaching module is created through two separate prompts—one for storytelling and another for activity generation—both centered around a single concept. This strategy follows the design principle of "divide and conquer", reducing cognitive load for the model and ensuring focus and depth at every step.

For each Core Concept in the knowledge base, the system first generates a micro-narrative using a carefully designed storytelling prompt. The prompt explicitly follows a three-part narrative structure: Problem → Solution → Impact, which aligns with Abrahamson's(1) narrative theory. In addition, elements from Alismail's(2) digital storytelling framework—such as Dramatic Question and Point of View—are embedded as storytelling hooks to increase emotional resonance and student engagement.

The prompt only includes data from one Core Concept at a time (e.g., its definition, significance, strengths, and weaknesses). This focused context allows the model to fully

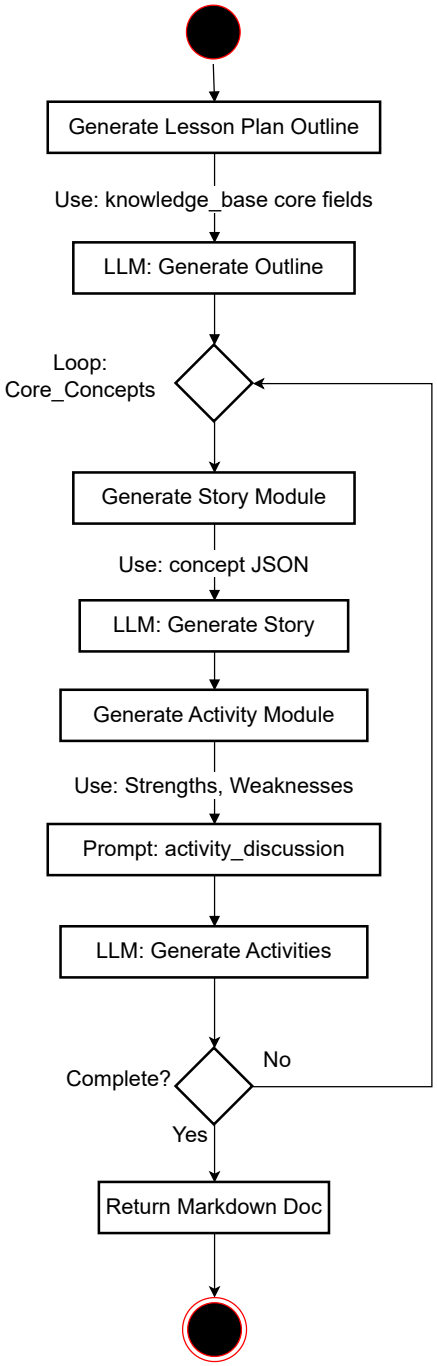

Figure 4.3: Flowchart of Story Generation and Narration Design

explore the concept, ensuring that the generated story is both detailed and educational. Each story is formatted in Markdown and labeled with a clear section title.

4.3.2 Narration and Interaction Design

Immediately after generating the story, the system proceeds to create interactive classroom activities for the same Core Concept. These activities are designed to reinforce key ideas and stimulate discussion, making the story more than just passive content.

The activity generation prompt asks the model to create:

- Debate questions based on strengths and weaknesses of the concept;
- Real-world scenarios where students must make decisions or solve problems.

This step operationalizes the concept's educational potential through interactive narration, following the principles of learner-centered pedagogy. Activities are designed to be adaptable for group discussions, in-class polls, or homework assignments.

Each activity section is generated separately and appended under its corresponding story module, forming a complete mini-package.

4.3.3 Lesson Guide and Classroom Use

Once both the story and its corresponding activity are generated for a concept, they are combined into one Markdown section. Over multiple iterations, this process builds a complete, modular lesson package, with each concept turned into an independent teaching module. These modules can be rearranged, removed, or reused depending on the teacher's needs.

This design brings three major benefits:

1. **Instructional clarity:** Each concept's story and activity are paired, making the material easy to teach and follow.
2. **Token efficiency:** Since each module is generated separately, there is no danger of running out of model context.
3. **Personalization potential:** Teachers can manually adjust individual modules or regenerate specific parts to suit different audiences.

Overall, the story generation and narration system bridges the gap between structured knowledge and educational storytelling. By adopting a modular, controllable design, teachers can flexibly guide the generation process while producing high-quality narrative outputs that are engaging, informative, and classroom-ready. The modular structure and interaction design further support meaningful learning experiences that extend beyond simple content delivery.

4.4 Dataset Construction

To support the development of a RAG-based data storytelling system in the educational domain, a high-quality dataset is essential. The goal of this dataset is to simulate realistic educational question-answer scenarios that align with actual teaching processes. We designed and generated a custom dataset tailored to classroom teaching and curriculum planning. The motivation for creating this dataset is threefold:

1. **Educational Relevance:** Existing datasets often lack structured teaching context or fail to align with knowledge points taught in actual courses. Our dataset directly reflects curriculum planning and instruction design around specific topics.

2. **Structured Knowledge Extraction:** Our system relies on the accurate identification of core concepts and their contextual importance. Therefore, our dataset is annotated with rich concept-level metadata for downstream reasoning and storytelling.

3. **Realistic Simulated Data:** Given the limited availability of labeled educational data, we simulate a teacher's thought process using GPT-4o to generate question-answer pairs that reflect real-world use cases automatically.

The dataset consists of 100 structured question-answer (QA) pairs, each centered around a specific knowledge topic, designed to simulate a teacher's lesson preparation process.

To generate realistic and domain-relevant content, we designed a semi-automated pipeline using GPT-4o. In this pipeline, the model was guided to act as a teacher preparing for a class session on a selected topic. The model received detailed instructions to simulate a planning query—typically representing a teacher requesting a breakdown of specific subtopics, conceptual comparisons, or instructional focus areas. These prompts closely mirror real-world teaching needs in computer science education.

For example, one such prompt was:

`{ "Question": "I need to prepare lessons on virtualization, with a focus on full virtualization, para-virtualization, and hardware-supported virtualization. Include how each method works, the role of hypervisors (Type 1 and Type 2), and performance implications." }`

From this input, the model generates a structured response that adheres to a strict schema, including the high-level **knowledge topic**, **core concepts** with detailed metadata, and a **concise summary**, as shown in Table 4.1. This schema design ensures consistency and completeness across all generated entries.

Field Name	Description
`Question`	The original user query simulating a teacher's request for preparing a lesson on a specific topic.
`Knowledge_Topic`	A high-level subject area that the question falls under, such as "Virtualization" or "Mutual Exclusion Algorithms"
`Core_Concepts`	A list of key concepts relevant to the question, each annotated with detailed semantic metadata (see below).
`Concept`	The name of the concept involved in the topic.
`Definition`	A concise explanation of the concept.
`Key_Points`	A list of major facts or ideas that define the concept, ideally three or more.
`Significance_Detail`	Explanation of why the concept is important or relevant in context.
`Strengths`	One or more strengths or benefits of the concept.
`Weaknesses`	One or more limitations or challenges associated with the concept.
`Source_Context`	Relevant background content supporting the question.
`Overall_Summary`	A one-to-two sentence summary that provides a direct, concise answer to the question.

Table 4.1: Schema of a QA Pair in the Dataset

While the topics are grounded in foundational and commonly taught areas of computer science, our focus is not on exhaustive topic coverage but rather on demonstrating the feasibility and effectiveness of structured QA generation for downstream storytelling tasks.

Importantly, the dataset was constructed with a forward-looking objective: to serve as

a foundational layer for generating rich, accurate, and pedagogically grounded educational stories. Each QA pair provides both the content and structure necessary for later stages of narrative construction with small language models, such as identifying causal relationships, sequencing ideas, and selecting relevant conceptual elements for data storytelling.

These datasets yield a total of 700 results in the RAG module (100 per language model) and a total of 4900 results in the Story Generation module.

4.5 Evaluation

To evaluate the system's generalizability and efficiency across different model scales and capabilities, we conducted a series of comparative experiments involving seven leading small language models: DeepSeek-LLM, Gemma, Qwen 2.5, OpenChat, LLaMA 3.1, OLMo 2, and Phi-4. These models were tested on identical input knowledge bases, using the same prompt templates and orchestration logic. We conducted a two-part evaluation focused on its two main functional stages: (1) Knowledge Extraction and (2) Story Generation and Narration.

4.5.1 Knowledge Extraction Evaluation

The goal of the Knowledge Extraction module is to convert educational PDF materials into a structured and machine-readable knowledge base in JSON format. The quality of this stage directly affects the reliability of all downstream story generation tasks. All evaluation metrics are scored on a scale from 0 (worst) to 5 (best).

We designed two automatic metrics to evaluate the knowledge extraction results:

1. **Retrieval Score:** This score measures how well the retrieved chunks semantically align with both the user's question and the key information in a reference set. For each reference sentence and the input question, we compute its maximum similarity with the retrieved chunks, and take the average across all.

2. **Generation Score:** This score assesses the quality of the generated answer, including faithfulness, accuracy, and completeness. We define three sub-criteria:

    - **Faithfulness:** We check whether the generated content is strictly grounded in the retrieved context.
    - **Accuracy:** This measures whether all key facts, concepts, and definitions are correct.

- **Completeness:** This evaluates whether all required aspects of the question are addressed. For example, in our dataset, the output should cover definition, significance, strengths, and weaknesses.

$$\text{Generation Score} = \frac{\text{Faithfulness} + \text{Accuracy} + \text{Completeness}}{3}$$

Together, these two scores provide a comprehensive view of the RAG system's performance. The Retrieval Score ensures that the system is using the right evidence, while the Generation Score evaluates whether the final output is both factually reliable and well-written. To reflect their relative importance, we compute an overall score as a weighted sum:

$$\text{Overall Score} = 0.4 * \text{Retrieval Score} + 0.6 * \text{Generation Score}$$

This weighting emphasizes the quality of the generated answer while still accounting for the relevance of the retrieved evidence. In many real-world scenarios, users primarily care about the final answer's fluency and factual accuracy, which often depends on—but is not solely determined by—the quality of retrieval.

4.5.2 Story Generation Evaluation

This stage is the most visible output of the system, where structured concepts are turned into engaging teaching stories and class activities. The evaluation here aims to assess the educational quality, logical consistency, and generation controllability across models.

We used two automatic evaluation metrics to score each generated module:

1. **Story Quality Score:** Each teaching story is evaluated by DeepSeek-V3 acting as a scorer. We defined a scoring rubric with four criteria:

   - **Structure:** Whether the story follows the required Problem → Solution → Impact structure
   - **Factual Accuracy:** Whether the story content stays consistent with the source concept
   - **Narrative Coherence:** Whether the story flows logically without contradictions

- **Educational Engagement:** Whether it includes teaching hooks like questions or analogies
- **Fluency & Consistency:** Whether the text contains grammar issues or hallucinations

$$\text{Story Quality Score} = \frac{\text{Sum of 5 Metrics}}{5}$$

2. **Activity Quality Score:** For each classroom activity (e.g., debate, roleplay, or scenario-based discussion) generated from the concept's strengths and weaknesses, we evaluate:

   - **Narrative Grounding:** Is the activity meaningfully embedded in the generated story context (e.g., characters, plot tension, or moral dilemma), rather than being a generic or disconnected task?
   - **Classroom Applicability Usability:** Can the activity be directly used in a classroom setting to promote student engagement, critical thinking, or collaborative exploration of the concept?

   $$\text{Activity Quality Score} = \frac{\text{Sum of 2 Metrics}}{2}$$

   Then we have:

$$\text{Overall Score} = \frac{\text{Activity Quality Score} + \text{Story Quality Score}}{2}$$

All evaluations are performed by prompting DeepSeek-V3 with carefully designed scoring rubrics. For each task (e.g., story quality, activity engagement), we define 3–4 detailed subcriteria. Each sub-score ranges from 0 (worst) to 5 (best). All final scores are calculated as the average of their respective criteria scores.

5

Results

This chapter presents a comprehensive evaluation of seven small language models in data storytelling for educational applications. The evaluation consists of two main components: the performance of the Knowledge Extraction, which is responsible for information retrieval and factual integration, and the performance of the Story Generation, which transforms retrieved information into coherent narrative texts. The evaluation follows a set of predefined criteria based on the proposed design in Section 4.5.

5.1 Knowledge Extraction Results

To better understand model behaviors in Knowledge Extraction tasks, we conducted a comparative evaluation across multiple models. The experimental results revealed notable differences in performance among the models. Their detailed evaluation scores are presented in the Figure 5.1, ranked in descending order by Overall RAG Score.

Key findings are summarized as follows:

1. **Overall Performance tiers among Models**

   The model Phi4_14b (3.67) achieved the highest overall score and outperformed others across all sub-metrics, establishing itself as the top performer. It was followed closely by Qwen2.5_7b (3.60). Together, these two models formed the top-performing group. In contrast, the remaining models belonged to a competitive middle tier, with minimal differences in their overall scores (only 0.07), indicating similar levels of competence in RAG tasks. Olmo2_7b (2.78) performed significantly worse than the others, with an overall score nearly 19% lower than that of the next-lowest model.

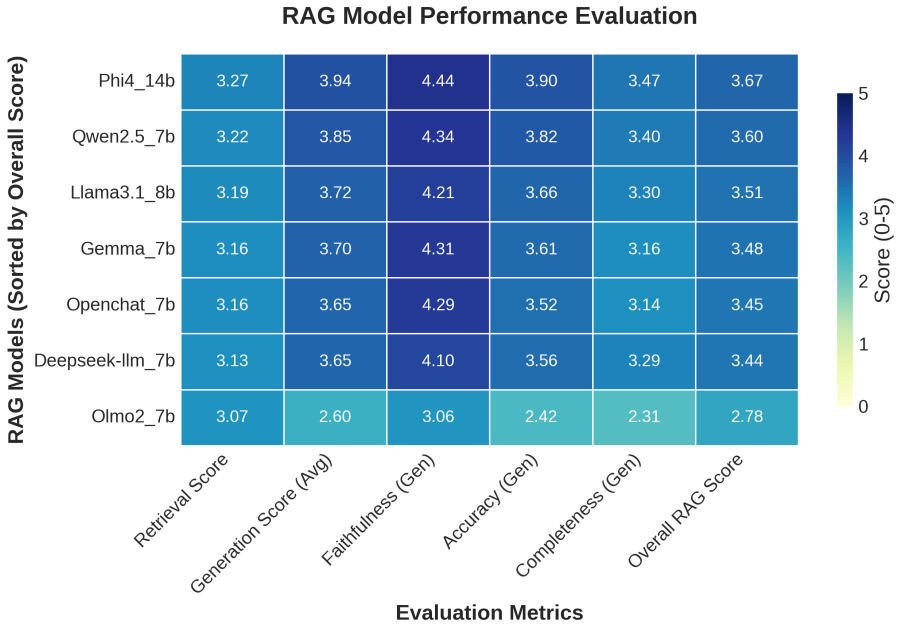

Figure 5.1: RAG Model Evaluation Results with All Metrics

2. **In-depth Dimension Analysis**

- **Retrieval Performance:** Differences in retrieval scores were relatively small. The top six models scored between 3.13 and 3.27, with minimal standard deviation. This suggests that the retrieval component of the RAG framework was stable and provided similarly relevant contexts across models. Even the worst-performing model, Olmo2_7b, achieved a score of 3.07, indicating that generation ability is the key factor determining final performance rather than retrieval quality.
- **Generation Performance**
  - **Faithfulness:** This was the strongest dimension for most models. The top six models all scored above 4.10, showing that SLMs can follow instructions and rely on retrieved content, effectively reducing hallucinated facts. Phi4_14b (4.44) performed exceptionally well in this aspect.
  - **Accuracy:** Performance differences became more apparent in this metric. Phi4_14b (3.90) and Qwen2.5_7b (3.82) could extract and express factual information more precisely, while Olmo2_7b (2.42) struggled significantly, often producing incorrect or imprecise answers.

- **Completeness:** This was the lowest-scoring and most discriminative submetric. Phi4_14b (3.47) again led in response comprehensiveness. However, models like Gemma_7b (3.16) and Openchat_7b (3.14) showed room for improvement, occasionally failing to provide fully complete answers.

Below is a comparison analysis of the knowledge extraction outputs from the Phi4_14b model (Output1) against the reference answers (Output2). While Output1 performs well in both retrieval and generation, the following differences are observed when compared to the reference output:

1. **Completeness** (score 4): Output1 covers the core concepts (Pods, Clusters, Master components, kubelets) but does not mention the two important background concepts of Container Orchestration and Orchestration Workflow. The standard answer provides a more comprehensive explanation of Kubernetes' context and overall workflow.

2. **Accuracy** (score: 5) and **Faithfulness** (core: 5): Both outputs accurately define the fundamental Kubernetes components and adhere closely to the source materials. However, the Master Components section in Output1 contains some redundancy, repeatedly emphasizing the maintenance of cluster state, whereas Output2 presents this information more succinctly using a tabular format to clearly delineate component functions.

3. Retrieval (score: 4): Output1 does not directly reference details from the source material about YAML/JSON configuration files and the orchestration workflow (e.g., "declarative deployment"), while Output2 clearly integrates these details, aligning more closely with the actual orchestration process.

Phi4_14b Output (Output1):

```
{
  "Question": "Create a
  ↪  narrative-style lesson on
  ↪  Kubernetes that introduces
  ↪  orchestration concepts such as
  ↪  Pods, Clusters, Master
  ↪  components, and kubelets,
  ↪  emphasizing how these elements
  ↪  scale microservice-based
  ↪  architectures.",
```

Reference Output (Output2):

```
{
    "Question": "Create a
    ↪  narrative-style lesson on
    ↪  Kubernetes that introduces
    ↪  orchestration concepts such as
    ↪  Pods, Clusters, Master
    ↪  components, and kubelets,
    ↪  emphasizing how these elements
    ↪  scale microservice-based
    ↪  architectures.",
```

```
"Knowledge_Topic": "Container
↪  Orchestration",
"Core_Concepts": [
  {
    "Concept": "Pods",
    "Definition": "A Pod is the
    ↪  smallest deployable unit in
    ↪  Kubernetes that can contain
    ↪  one or more containers
    ↪  sharing storage and network
    ↪  resources.",
    "Key_Points": [
      "Pods allow multiple
      ↪  containers to be deployed
      ↪  together, sharing
      ↪  resources like networking
      ↪  and storage.",
      "They are managed by
      ↪  Kubernetes as a single
      ↪  entity.",
      "Pods facilitate the
      ↪  deployment of
      ↪  microservices within a
      ↪  containerized
      ↪  environment."
    ],
    "Significance_Detail": "Pods are
    ↪  crucial for managing the
    ↪  lifecycle of containers in
    ↪  Kubernetes, enabling
    ↪  efficient scaling and
    ↪  management of
    ↪  microservice-based
    ↪  architectures.",
    "Strengths": "Pods simplify the
    ↪  deployment process by
    ↪  grouping related containers
    ↪  together, making it easier
    ↪  to manage their lifecycle
    ↪  and resources.",
    "Weaknesses": null
  },
  {
```

```
"Knowledge_Topic": "Container
↪  Orchestration",
"Core_Concepts": [
    {
        "Concept": "Container
        ↪  Orchestration",
        "Definition": "The
        ↪  automation of the
        ↪  deployment,
        ↪  management, scaling,
        ↪  and networking of
        ↪  containers, providing
        ↪  a framework for
        ↪  managing containerized
        ↪  applications at
        ↪  scale.",
        "Key_Points": [
            "It automates the
            ↪  entire lifecycle
            ↪  of containers.",
            "It is beneficial for
            ↪  enterprises
            ↪  managing hundreds
            ↪  or thousands of
            ↪  containers.",
            "Orchestration tools
            ↪  allow the
            ↪  deployment of the
            ↪  same application
            ↪  across different
            ↪  environments
            ↪  without
            ↪  redesign.",
            "It supports DevOps
            ↪  teams by
            ↪  integrating into
            ↪  CI/CD workflows.",
            "Enables running
            ↪  multiple app parts
            ↪  independently on
            ↪  shared hardware.",
```

```
    "Concept": "Clusters",
    "Definition": "A Cluster in
    ↪ Kubernetes is a group of
    ↪ nodes that work together to
    ↪ run containerized
    ↪ applications.",
    "Key_Points": [
      "Clusters can span across
      ↪ public, private, or hybrid
      ↪ clouds.",
      "They provide the
      ↪ infrastructure necessary
      ↪ for running and managing
      ↪ containers at scale.",
      "Kubernetes clusters
      ↪ facilitate rapid scaling
      ↪ and workload portability."
    ],
    "Significance_Detail": "Clusters
    ↪ are essential for hosting
    ↪ cloud-native applications
    ↪ that require scalability and
    ↪ flexibility across different
    ↪ environments.",
    "Strengths": "Clusters enable
    ↪ Kubernetes to manage
    ↪ large-scale containerized
    ↪ workloads efficiently,
    ↪ supporting both on-premise
    ↪ and cloud deployments.",
    "Weaknesses": null
  },
  {
    "Concept": "Master Components",
    "Definition": "The Master
    ↪ components in Kubernetes are
    ↪ responsible for managing the
    ↪ state of the cluster and
    ↪ making global decisions
    ↪ about the cluster.",
    "Key_Points": [
      "Uses YAML/JSON
      ↪ configuration
      ↪ files to define
      ↪ application
      ↪ setups."
    ],
    "Significance_Detail":
    ↪ "Container
    ↪ orchestration solves
    ↪ the complexity of
    ↪ managing a large
    ↪ number of containers
    ↪ manually. It is the
    ↪ foundation for running
    ↪ scalable, resilient
    ↪ microservice-based
    ↪ applications in
    ↪ production
    ↪ environments.",
    "Strengths": "Enables
    ↪ automated scaling,
    ↪ management, and
    ↪ deployment of
    ↪ containerized
    ↪ applications,
    ↪ supporting resilient
    ↪ and portable
    ↪ microservices
    ↪ architectures.",
    "Weaknesses": null
  },
  {
    "Concept": "Kubernetes",
    "Definition": "An
    ↪ open-source container
    ↪ orchestration tool,
    ↪ originally developed
    ↪ by Google, that
    ↪ automates the
    ↪ deployment, scaling,
    ↪ and management of
    ↪ containerized
    ↪ applications.",
```

```
      "The Master node controls the
      ↪  scheduling, scaling, and
      ↪  health management of
      ↪  containers.",
      "It includes components like
      ↪  the API server, scheduler,
      ↪  and controller manager.",
      "These components ensure that
      ↪  the desired state of the
      ↪  cluster is maintained."
    ],
    "Significance_Detail": "Master
    ↪  components are vital for
    ↪  orchestrating container
    ↪  operations across the entire
    ↪  Kubernetes cluster, ensuring
    ↪  efficient resource
    ↪  utilization and application
    ↪  reliability.",
    "Strengths": "They provide
    ↪  centralized control over the
    ↪  cluster, enabling consistent
    ↪  management and
    ↪  decision-making processes.",
    "Weaknesses": null
  },
  {
    "Concept": "Kubelets",
    "Definition": "Kubelets are
    ↪  agents that run on each node
    ↪  in a Kubernetes cluster,
    ↪  responsible for maintaining
    ↪  the desired state of
    ↪  containers.",
    "Key_Points": [
      "Kubelets communicate with the
      ↪  Master components to
      ↪  receive instructions.",
      "They manage the lifecycle of
      ↪  containers on their
      ↪  respective nodes.",
    "Key_Points": [
      "It was donated by
      ↪  Google to the
      ↪  Cloud-Native
      ↪  Computing
      ↪  Foundation (CNCF)
      ↪  in 2015.",
      "It allows you to
      ↪  build application
      ↪  services that span
      ↪  multiple
      ↪  containers and
      ↪  schedule them
      ↪  across a
      ↪  cluster.",
      "Kubernetes eliminates
      ↪  many manual
      ↪  processes involved
      ↪  in deploying and
      ↪  scaling
      ↪  applications.",
      "Supports workload
      ↪  portability and
      ↪  load balancing
      ↪  without
      ↪  redesigning
      ↪  applications.",
      "It can manage
      ↪  clusters spanning
      ↪  public, private,
      ↪  or hybrid clouds,
      ↪  making it ideal
      ↪  for cloud-native
      ↪  apps."
    ],
```

```
      "Kubelets ensure that
      ↪ containers are running as
      ↪ expected and report back
      ↪ to the Master."
    ],
    "Significance_Detail": "Kubelets
    ↪ play a critical role in
    ↪ executing container
    ↪ orchestration tasks at the
    ↪ node level, ensuring that
    ↪ applications run smoothly
    ↪ across the cluster.",
    "Strengths": "They enable
    ↪ decentralized management of
    ↪ containers, allowing for
    ↪ efficient scaling and
    ↪ resource allocation.",
    "Weaknesses": null
  }
],
"Overall_Summary": "Kubernetes
↪ orchestrates containerized
↪ microservices through Pods,
↪ Clusters, Master components, and
↪ kubelets. These elements work
↪ together to scale applications
↪ efficiently across diverse
↪ environments, providing robust
↪ management of resources and
↪ application lifecycles."
"Source_Context": [
  {
    "source": ".../lecture.md",
    "content_type": "normal",
    "Significance_Detail":
    ↪ "Kubernetes has become
    ↪ the de facto standard
    ↪ for container
    ↪ orchestration,
    ↪ providing a powerful
    ↪ and flexible platform
    ↪ to fully implement and
    ↪ rely on a
    ↪ container-based
    ↪ infrastructure in
    ↪ production.",
    "Strengths": "Provides
    ↪ workload portability,
    ↪ load balancing, and
    ↪ automated management,
    ↪ helping to implement
    ↪ resilient
    ↪ container-based
    ↪ infrastructure.",
    "Weaknesses": null
  },
  {
    "Concept": "Kubernetes
    ↪ Components",
    "Definition": "The
    ↪ fundamental building
    ↪ blocks of a Kubernetes
    ↪ system that work
    ↪ together to manage a
    ↪ containerized
    ↪ application.",
    "Key_Points": [
      "Cluster: A group of
      ↪ nodes, consisting
      ↪ of at least one
      ↪ master node and
      ↪ several worker
      ↪ nodes.",
```

```
    "page_content": "Kubernetes is
    ↪  an open source container
    ↪  orchestration tool that was
    ↪  originally developed and
    ↪  designed by engineers at
    ↪  Google. Google donated the
    ↪  Kubernetes project to the
    ↪  newly formed Cloud-Native
    ↪  Computing Foundation in
    ↪  2015. Kubernetes
    ↪  orchestration allows you to
    ↪  build application services
    ↪  that span multiple
    ↪  containers, schedule
    ↪  containers across a cluster,
    ↪  scale those containers, and
    ↪  manage their health over
    ↪  time. Kubernetes eliminates
    ↪  many of the manual processes
    ↪  involved in deploying and
    ↪  scaling"
  },
  {
    "source": ".../lecture.md",
    "content_type": "table",
    "page_content": "Master Node: The
    ↪  machine that
    ↪  controls the
    ↪  Kubernetes nodes
    ↪  and where all task
    ↪  assignments
    ↪  originate.",
    "Kubelet: A service
    ↪  that runs on each
    ↪  worker node, reads
    ↪  container
    ↪  manifests, and
    ↪  ensures the
    ↪  defined containers
    ↪  are started and
    ↪  running.",
    "Pod: The smallest
    ↪  deployable unit in
    ↪  Kubernetes,
    ↪  representing a
    ↪  group of one or
    ↪  more containers
    ↪  that are deployed
    ↪  to a single node
    ↪  and share
    ↪  resources like an
    ↪  IP address and
    ↪  hostname."
  ],
```

```
  "page_content": "and rely on a
  ↪ container-based
  ↪ infrastructure in production
  ↪ environments. These clusters
  ↪ can span hosts across
  ↪ public, private, or hybrid
  ↪ Clouds. For this reason,
  ↪ Kubernetes is an ideal
  ↪ platform for hosting
  ↪ Cloud-native apps that
  ↪ require rapid scaling.
  ↪ Kubernetes also assists with
  ↪ workload portability and
  ↪ load balancing by letting
  ↪ you move applications
  ↪ without redesigning them.
  ↪ <html><body><table><tr><td>
  ↪ Kubernetes components
  ↪ </td><td> Description
  ↪ </td></tr><tr><td> Cluster
  ↪ </td><td> A group of nodes,
  ↪ with at"
},
{
  "source": ".../lecture.md",
  "content_type": "normal",
  "page_content": "Container
  ↪ orchestration tools provide
  ↪ a framework for managing
  ↪ containers and microservices
  ↪ architecture at scale. Many
  ↪ container orchestration
  ↪ tools that can be used for
  ↪ container lifecycle
  ↪ management. Some popular
  ↪ options are Kubernetes,
  ↪ Docker Swarm, and Apache
  ↪ Mesos. # 5.1 Kubernetes"
},
{
  "source": ".../lecture.md",
  "content_type": "normal",
  "Significance_Detail":
  ↪ "Understanding these
  ↪ components is
  ↪ essential for
  ↪ operating a Kubernetes
  ↪ cluster. The master
  ↪ node acts as the
  ↪ brain, the worker
  ↪ nodes provide the
  ↪ brawn (compute
  ↪ resources), the
  ↪ kubelet is the agent
  ↪ on each node, and the
  ↪ pod is the basic unit
  ↪ that encapsulates the
  ↪ application
  ↪ containers.",
  "Strengths": "Provides
  ↪ robust management of
  ↪ containerized
  ↪ workloads.",
  "Weaknesses": null
},
{
  "Concept": "Orchestration
  ↪ Workflow",
  "Definition": "The process
  ↪ by which a container
  ↪ orchestration tool
  ↪ like Kubernetes
  ↪ deploys and manages an
  ↪ application.",
  "Key_Points": [
    "A user describes the
    ↪ desired
    ↪ application
    ↪ configuration in a
    ↪ YAML or JSON
    ↪ file.",
```

```
    "page_content": "were determined
    ↪  in the compose file. You can
    ↪  use Kubernetes patterns53 to
    ↪  manage the configuration,
    ↪  lifecyle, and scale of
    ↪  containerbased applications
    ↪  and services. These
    ↪  repeatable patterns are the
    ↪  tools needed by a Kubernetes
    ↪  developer to build complete
    ↪  systems. Container
    ↪  orchestration can be used in
    ↪  any environment that runs
    ↪  containers, including
    ↪  onpremise servers and
    ↪  public"
  },
  {
    "source": ".../lecture.md",
    "content_type": "normal",
    "page_content": "‘‘Container
    ↪  orchestration automates the
    ↪  deployment, management,
    ↪  scaling, and networking of
    ↪  containers. Enterprises that
    ↪  need to deploy and manage
    ↪  hundreds or thousands of
    ↪  containers can benefit from
    ↪  container orchestration.
    ↪  Containers orchestration can
    ↪  help you to deploy the same
    ↪  application across different
    ↪  environments without needing
    ↪  to redesign it. And
    ↪  microservices in containers
    ↪  make it easier to
    ↪  orchestrate services,
    ↪  including storage,
    ↪  networking, and security.
    ↪  Containers give your
    ↪  microservice-based"
  }
]
                "This file specifies
                ↪  container images,
                ↪  networking rules,
                ↪  and log storage
                ↪  locations.",
                "The orchestration
                ↪  tool automatically
                ↪  schedules the
                ↪  deployment to a
                ↪  cluster, finding a
                ↪  suitable host.",
                "It then manages the
                ↪  container's
                ↪  lifecycle based on
                ↪  the specifications
                ↪  in the
                ↪  configuration
                ↪  file."
            ],
            "Significance_Detail":
            ↪  "This declarative
            ↪  approach allows
            ↪  developers to define
            ↪  the 'what' (the
            ↪  desired state of the
            ↪  application) and lets
            ↪  the orchestration tool
            ↪  handle the 'how' (the
            ↪  steps to achieve and
            ↪  maintain that state),
            ↪  simplifying complex
            ↪  deployments and
            ↪  ensuring
            ↪  consistency.",
            "Strengths": null,
            "Weaknesses": null
        }
    ],
```

```
}
```

```
"Overall_Summary": "Imagine you
↪  have an application made of
↪  many small microservices, each
↪  in its own container. Managing
↪  hundreds of these manually is
↪  impossible. This is where
↪  Kubernetes, the conductor of
↪  our container orchestra, comes
↪  in. You start by creating a
↪  'Cluster,' which is your
↪  orchestra pit-a group of
↪  computers (nodes). The 'Master
↪  Node' is the conductor; it
↪  decides which instrument
↪  (container) plays where. On
↪  each musician's stand (a
↪  worker node), there's a
↪  'Kubelet,' an agent that makes
↪  sure the right containers are
↪  running. Your application's
↪  containers don't just run
↪  loose; they are grouped into
↪  'Pods,' the smallest unit in
↪  Kubernetes, like a section of
↪  the orchestra (e.g., the
↪  strings). You, the composer,
↪  write a score (a YAML file)
↪  describing how your
↪  application should run.
↪  Kubernetes reads this score
↪  and automatically handles
↪  deploying, scaling, and
↪  managing the pods, ensuring
↪  your microservices application
↪  runs smoothly and resiliently
↪  at any scale."
"Source_Context": [
    {
        "source":
        ↪  ".../lecture.md",
        "content_type": "normal",
```

```
        "page_content":
        ↪  "‘‘Container
        ↪  orchestration
        ↪  automates the
        ↪  deployment,
        ↪  management, scaling,
        ↪  and networking of
        ↪  containers.
        ↪  Enterprises that need
        ↪  to deploy and manage
        ↪  hundreds or thousands
        ↪  of containers can
        ↪  benefit from container
        ↪  orchestration.
        ↪  Containers
        ↪  orchestration can help
        ↪  you to deploy the same
        ↪  application across
        ↪  different environments
        ↪  without needing to
        ↪  redesign it. And
        ↪  microservices in
        ↪  containers make it
        ↪  easier to orchestrate
        ↪  services, including
        ↪  storage, networking,
        ↪  and security."
    },
    {
        "source":
        ↪  ".../lecture.md",
        "content_type": "normal",
```

```
        "page_content":
        ↪  "Kubernetes is an open
        ↪  source container
        ↪  orchestration tool
        ↪  that was originally
        ↪  developed and designed
        ↪  by engineers at
        ↪  Google. Google donated
        ↪  the Kubernetes project
        ↪  to the newly formed
        ↪  Cloud-Native Computing
        ↪  Foundation in
        ↪  2015.\n\nKubernetes
        ↪  orchestration allows
        ↪  you to build
        ↪  application services
        ↪  that span multiple
        ↪  containers, schedule
        ↪  containers across a
        ↪  cluster, scale those
        ↪  containers, and manage
        ↪  their health over
        ↪  time."
    },
    {
        "source":
        ↪  ".../lecture.md",
        "content_type": "normal",
```

```
    "page_content":
    ↪  "<html><body><table><tr><td>
    ↪  Kubernetes components
    ↪  </td><td> Description
    ↪  </td></tr><tr><td>
    ↪  Cluster </td><td> A
    ↪  group of nodes, with
    ↪  at least one master
    ↪  node and several
    ↪  worker
    ↪  nodes.</td></tr><tr><td>
    ↪  Master </td><td> The
    ↪  machine that controls
    ↪  Kubernetes nodes. This
    ↪  is where all task
    ↪  assignments
    ↪  originate.</td></tr><tr><td>
    ↪  Kubelet </td><td> This
    ↪  service runs on nodes
    ↪  and reads the
    ↪  container manifests
    ↪  and ensures the
    ↪  defined containers are
    ↪  started and
    ↪  running.</td></tr><tr><td>
    ↪  Pod </td><td>. A group
    ↪  of one or more
    ↪  containers deployed to
    ↪  a single node. All
    ↪  containers in a pod
    ↪  share an IP address,
    ↪  IPC, hostname,and
    ↪  other resources.
    ↪  </td></tr></table></body></html>"
},
{
    "source":
    ↪  ".../lecture.md",
    "content_type": "normal",
```

```
                "page_content": "When you
                ↪  use a container
                ↪  orchestration tool,
                ↪  such as Kubernetes,
                ↪  you will describe the
                ↪  configuration of an
                ↪  application using
                ↪  either a YAML or JSON
                ↪  file. The
                ↪  configuration file
                ↪  tells the
                ↪  configuration
                ↪  management tool where
                ↪  to find the container
                ↪  images, how to
                ↪  establish a network,
                ↪  and where to store
                ↪  logs.\n\nWhen
                ↪  deploying a new
                ↪  container, the
                ↪  container management
                ↪  tool automatically
                ↪  schedules the
                ↪  deployment to a
                ↪  cluster and finds the
                ↪  right host,
                ↪  considering any
                ↪  defined requirements
                ↪  or restrictions. The
                ↪  orchestration tool
                ↪  then manages the
                ↪  container's lifecycle
                ↪  based on the
                ↪  specifications that
                ↪  were determined in the
                ↪  compose file."
            }
        ]
    }
```

5.2 Story Generation Results

This section evaluates the overall performance of different language model combinations on two tasks: story generation and the generation of corresponding interactive narration. A total of 49 unique model combinations were tested. Detailed evaluation scores are shown in the Figure 5.2, ranked in descending order by Overall Score.

Key findings are summarized as follows:

1. **Performance Tiers Analysis**

   - **Top-Tier Performers (Score: 3.70 - 3.83):** This tier is dominated by model combinations that use Phi4_14b as the RAG model. The top three combinations, Phi4_14b + Qwen2.5_7b (3.83), Phi4_14b + Olmo2_7b (3.79), and Phi4_14b + Llama3.1_8b (3.77), demonstrate the crucial role of high-quality context in generating top-level educational content. Qwen2.5_7b also showed strong performance as a story generator, appearing in four of the top six combinations, indicating its outstanding narrative capabilities. An example is analysed below.
   - **Competitive Middle-Tier (Score: 3.40 - 3.70):** Combinations in this tier generally perform well across all metrics but lack exceptional strengths. A clear trend is observed: when Qwen2.5_7b, Llama3.1_8b, or Olmo2_7b serve as story generators, the combinations tend to fall in the upper range of this tier. In contrast, combinations with Openchat_7b or Deepseek-llm_7b as story generators tend to score in the lower part of this range. The example (Phi4_14b + Openchat_7b) is provided in Appendix A.1.
   - **Underperformers (Score: below 3.40):** Model combinations in this tier show noticeable weaknesses across multiple dimensions. Surprisingly, most combinations using Phi4_14b or Gemma_7b as story generators fall into this category. For example, Phi4_14b + Phi4_14b scored only 3.28 overall, with a particularly low score of 1.86 in "Educational Engagement", revealing major deficiencies in creativity and emotional connection. The example (Phi4_14b + Phi4_14b) is provided in Appendix A.2.
   - **Lagging Performers (Score: below 2.0):** All combinations where Olmo2_7b was used as the RAG model fell into this lowest tier, with the worst score being 1.28. This suggests that Olmo2_7b provides poor-quality context or lacks

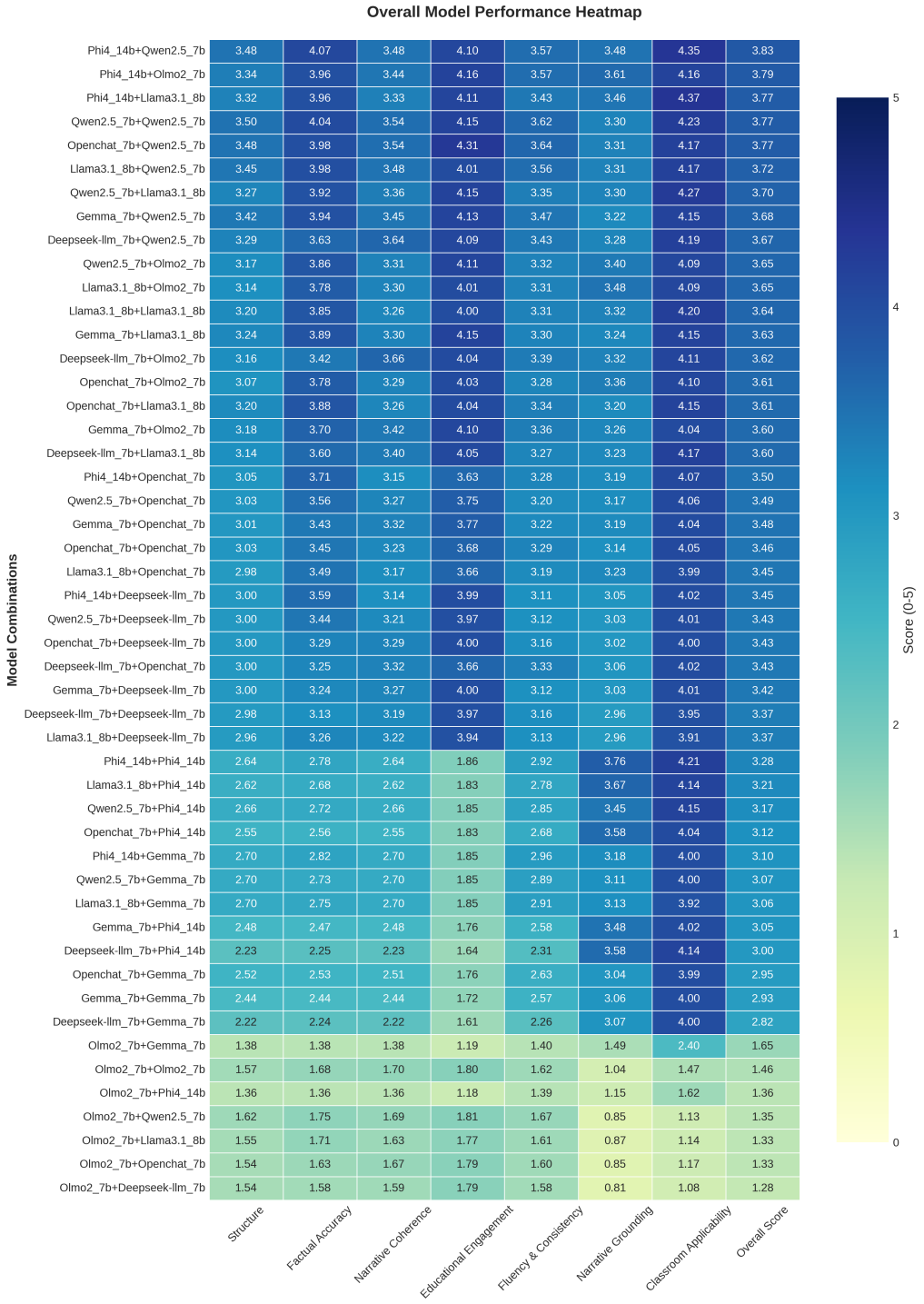

Figure 5.2: Story Generation Evaluation Results Across 49 Combinations of Extraction Models and Generation Models

alignment with the downstream task, which severely impacts the final content quality, regardless of the strength of the story generator.

2. **In-depth Dimension Analysis**

- **Story Quality**
  - **Structure & Narrative Coherence:** These two closely related dimensions are primarily influenced by the story generator. Qwen2.5_7b performs strongly here, producing well-structured and coherent stories. In contrast, when Phi4_14b or Gemma_7b is used as the generator, their outputs often lack clear logic and flow, which contributes significantly to their lower overall scores.
  - **Factual Accuracy:** Combinations with Phi4_14b as the RAG model consistently score the highest in factual accuracy (mostly above 3.9), showing that high-quality source content is essential for producing faithful output. On the other hand, when Olmo2_7b serves as the RAG model, all its combinations perform poorly in this dimension (often below 1.8), proving that the quality of its source concepts is the key bottleneck in the whole pipeline.
  - **Educational Engagement:** The ability to embed teaching hooks (such as questions or analogies) varies greatly across different story generators. Models like Qwen2.5_7b, Llama3.1_8b, and Olmo2_7b show strong capabilities in this area, frequently scoring above 4.1. One of the most critical findings in this evaluation is that Phi4_14b, when used as a story generator, consistently fails on this task, scoring below 2.0. This suggests a significant weakness in executing instructions that involve embedding educational elements.
  - **Fluency & Consistency:** Qwen2.5_7b stands out, regularly generating logically consistent and grammatically fluent stories (both scores above 3.4), except the combination with Olmo_7b.
- **Activity Quality**
  - **Narrative Grounding:** This metric reveals an interesting pattern. Although Phi4_14b tends to generate less engaging stories when used as a story generator, the activities it designs are often highly aligned with the story content (Combinations: Phi4_14b + Phi4_14b (3.76), Llama3.1_8b + Phi4_14b (3.67) and Phi4_14b + Olmo2_7b (3.61)). This suggests that

simpler, more structured stories might actually make it easier to create logically consistent follow-up tasks.

- **Classroom Applicability:** This metric is contributed by strong content quality (ensured by the Knowledge Extraction) and effective instructional design (realized by the story generator). Phi4_14b, as the RAG model, provided accurate and reliable conceptual foundations. Meanwhile, Llama3.1_8b and Qwen2.5_7b, serving as story generators, contributed creative and practical instructional designs that were engaging, actionable, and pedagogically meaningful.

Below is the analysis of the story generation outputs from the Qwen2.5_7b model (based on the knowledge extraction output from Phi4_14b):

1. **Story Quality Score**

   - **Structure Integrity (score: 4)**
     - **Strengths:** Each module (Pods, Clusters, Master Components, Kubelets) follows a three-part structure: *Problem* → *Solution* → *Impact*, making the logic clear.
     - **Improvements:** Some modules (e.g., Clusters) could have more specific *Impact* sections, such as adding real-world cases to enhance persuasiveness.
   - **Factual Accuracy (score: 5)** All technical definitions align perfectly with the knowledge extraction result, with no errors.
   - **Narrative Coherence (score: 4)**
     - **Strengths:** The analogies like "city management" and "orchestra conductor" are consistently used, keeping the theme unified.
     - **Improvements:** The connection between Master Components and Kubelets could be emphasized more to avoid a fragmented feel between modules.
   - **Educational Engagement (score: 4)**
     - **Highlights:** Scenario-based questions (e.g., "How to handle millions of users?") and analogies (e.g., "Pods are families sharing resources") effectively stimulate interest.
     - **Suggestions:** Adding more interactive questions (e.g., "What happens if a container in a Pod crashes?") could increase student participation.

- **Fluency and Consistency (score: 5)** The language is smooth, balancing technical terms and plain explanations well, suitable for beginners.

2. **Activity Quality Score**

- **Narrative Grounding (score: 4)**
  - **Strengths:** The debates and scenario questions closely align with the module themes.
  - **Improvements:** Activities could more directly reference the analogies from the story (e.g., having students use the "orchestra conductor" analogy to explain the role of the Master components).
- **Classroom Applicability (score: 5)** The debate design is actionable, with clear pro and con arguments. Scenario questions are close to real development scenarios.

Story Generation from Qwen2.5_7b:

```
```markdown
# Lesson Title: Scaling Microservices with Kubernetes: An Introduction to
↪  Container Orchestration

## Introduction (Hook)
Objective: To grab students' attention by posing a real-world problem about
↪  managing containerized microservices at scale.

**Scenario**: Imagine you are developing an e-commerce platform that needs to
↪  handle millions of users simultaneously. How would you ensure your
↪  application remains robust and performs well under such demands? Today, we
↪  will explore how Kubernetes can help solve this challenge through its
↪  powerful orchestration capabilities.

## Core Content Delivery
1. **Understanding Pods**
   Objective: To define what a Pod is in the context of Kubernetes and explain
   ↪  why it's crucial for microservice management.
2. **Introduction to Clusters**
   Objective: To introduce students to the concept of Kubernetes clusters and how
   ↪  they form the backbone of container orchestration.
3. **Master Components Overview**
   Objective: To provide an overview of key master components in a Kubernetes
   ↪  cluster, their roles, and why they are essential.
```
```

```
4. **Role of Kubelets**
   Objective: To explain the role and responsibilities of kubelets within a
   ↪  Kubernetes cluster.

## Key Activity/Discussion
Objective: An interactive segment to reinforce learning through real-world
↪  examples or case studies.

**Activity**: Divide students into small groups and provide them with scenarios
↪  where they need to design a microservice-based application using Kubernetes.
↪  Each group will present their solution, focusing on the role of Pods,
↪  Clusters, Master Components, and Kubelets in managing their services.

## Conclusion & Synthesis
Objective: To wrap up the lesson by revisiting the original question and
↪  connecting back to the Overall Summary.

**Summary**: By understanding how Pods, Clusters, Master Components, and Kubelets
↪  work together, you can effectively scale microservice-based architectures.
↪  This knowledge is crucial for managing complex applications that require
↪  robust resource management and lifecycle control.
```

```
## Teaching Module: Pods
### The Story (Problem -> Solution -> Impact)

#### The Problem (Event)
In the world of software development and deployment, managing multiple services
↪  that work together can be incredibly complex. Imagine you're an engineer
↪  tasked with deploying a web application where different parts of the system
↪  need to communicate seamlessly. You have databases, web servers, and various
↪  microservices all running in separate containers. Without a cohesive
↪  approach, ensuring these components work together smoothly becomes a daunting
↪  task.

#### The 'Aha!' Moment (Experience)
```

```
Enter Pods! Kubernetes, the popular open-source platform for automating container
↪  operations, introduces Pods as a solution to this problem. A Pod is like a
↪  cozy little house where multiple containers live together, sharing resources
↪  and working in harmony. Think of it as a family home where siblings share a
↪  kitchen, bathroom, and living room but still have their own rooms.

- **Pods allow multiple containers to be deployed together, sharing resources
↪  like networking and storage**.
- **They are managed by Kubernetes as a single entity**, making sure everything
↪  works smoothly even when one part of the household (container) needs
↪  attention.
- **Pods facilitate the deployment of microservices within a containerized
↪  environment**, ensuring that all components work cohesively.

#### The Impact (Meaning)
The introduction of Pods into the world of Kubernetes has revolutionized how we
↪  manage and deploy complex applications. By grouping related containers
↪  together, Pods simplify the deployment process and make it easier to manage
↪  their lifecycle and resources. This means you can focus more on writing great
↪  code rather than worrying about the nitty-gritty details of container
↪  management.

### Storytelling Hooks

#### Dramatic Question
Could making a computer dumber actually make it smarter? In the case of Pods,
↪  it's true-by creating a simpler system to manage containers, Kubernetes makes
↪  the overall process much more efficient and manageable.

#### Point of View
From the perspective of an engineer facing the challenge of deploying a complex
↪  application with multiple microservices, Pods provide a powerful tool for
↪  managing these components effectively. Imagine having a set of keys that can
↪  unlock a world where your containers work together seamlessly-Pods are like
↪  those keys.

### Classroom Delivery Tips

#### Pacing
- **Pause after explaining what Pods are** to give students time to absorb the
↪  concept.
- **Ask a question**: "Can you think of any situations where grouping related
↪  containers together would be beneficial?"
```

```
#### Analogy
To help students understand the idea, use an analogy: "Imagine you're planning a
↪  party and need to set up a table with multiple dishes. Instead of setting
↪  each dish on separate tables (containers), why not place them all on one
↪  large table (Pod)? This way, everything is neatly organized, and guests can
↪  access all the food easily."

By weaving these elements together, the concept of Pods becomes engaging and easy
↪  to understand for students, helping them grasp its importance in managing
↪  microservices effectively.

### Interactive Activities for Pods
### Debate Topic:
**Resolved: The benefits of using Pods in container orchestration outweigh any
↪  potential drawbacks.**

#### Proponents (Affirmative):
- Pods simplify the deployment process by grouping related containers, making it
↪  easier for administrators to manage a set of containers as a single entity.
- They streamline resource allocation and lifecycle management, ensuring that
↪  containers within a pod share network namespaces and security contexts.

#### Opponents (Negative):
- There are no significant weaknesses mentioned in the strengths provided.
↪  Therefore, this side would argue that the debate is based solely on
↪  theoretical constructs without concrete evidence of drawbacks.
- The absence of identified weaknesses makes it challenging to present valid
↪  counterarguments against the use of Pods.

### What If Scenario Question:
**What if a software development team at your university decided to migrate their
↪  application from traditional server deployment to container orchestration
↪  using Kubernetes, but they are debating whether or not to use Pods for
↪  grouping their containers?**

**Scenario:**
Your team is developing a new web application that consists of multiple services
↪  such as the front-end interface, back-end logic, and database. The team has
↪  decided to containerize these components and deploy them in a Kubernetes
↪  cluster. However, they are divided on whether or not to use Pods for grouping
↪  their containers.
```

```
**Question:**
Given the strengths of Pods mentioned (simplifying deployment, easier lifecycle
↪  management), as well as the absence of any weaknesses, how would you advise
↪  the team to proceed with their containerization strategy? Justify your
↪  recommendation by explaining why using Pods could be beneficial in this
↪  scenario and what trade-offs they should consider.

**Expected Student Response:**
- Advise the use of Pods due to their simplifying nature for deployment and
↪  management.
- Highlight that pods can help manage network resources more efficiently,
↪  ensuring better performance.
- Suggest considering the complexity introduced by managing multiple pods if
↪  there are a large number of related containers.
- Recommend using pods where necessary but also planning for scenarios where
↪  single containers might be deployed without them to maintain flexibility.


---

## Teaching Module: Clusters
### The Story (Problem -> Solution -> Impact)

---

#### The Problem (Event)
Imagine you're an engineer tasked with managing a fleet of microservices
↪  applications that your company has recently developed. These services are
↪  complex and require high availability, scalability, and reliability. Each
↪  service is built using containerized applications, but manually deploying and
↪  managing these containers across multiple servers in different
↪  environments-both on-premise and public clouds-is becoming increasingly
↪  cumbersome. The challenge lies not just in the manual effort required to
↪  manage so many containers but also ensuring that your applications can
↪  seamlessly scale up or down based on demand.

#### The 'Aha!' Moment (Experience)
```

```
One day, you stumble upon a platform called Kubernetes, which promises a solution
↪  for these challenges. As you dive deeper into its documentation and community
↪  forums, you realize that the key to this problem lies in something called
↪  "Clusters." A Cluster, as defined by Kubernetes, is a group of nodes that
↪  work together to run containerized applications. You learn that these
↪  clusters can span across public, private, or hybrid clouds, providing a
↪  flexible infrastructure for running containers at scale.

To understand how it works, you start with the three key points:
1. Clusters can indeed be spread across different cloud environments, offering
↪  flexibility.
2. They provide necessary infrastructure for managing large-scale containerized
↪  workloads efficiently.
3. Kubernetes clusters facilitate rapid scaling and workload portability,
↪  ensuring that your applications can dynamically adjust to changing loads.

You experiment by setting up a small cluster on a virtual machine in the cloud,
↪  deploying some of your microservices, and observing how they automatically
↪  scale based on CPU usage or other metrics. The "Aha!" moment comes when you
↪  see not only how seamlessly these services operate but also how easy it is to
↪  manage them using Kubernetes commands.

#### The Impact (Meaning)
The impact of this solution is profound. Clusters enable Kubernetes to manage
↪  large-scale containerized workloads efficiently, supporting both on-premise
↪  and cloud deployments. This means that your applications can now be deployed
↪  with ease across different environments, ensuring high availability and
↪  reliability. Moreover, the ability to quickly scale up or down based on
↪  demand significantly reduces costs while maintaining optimal performance.

From a technical standpoint, clusters provide the infrastructure necessary for
↪  running and managing containers at scale, which is essential for hosting
↪  cloud-native applications that require scalability and flexibility. This not
↪  only makes your job as an engineer easier but also allows the company to
↪  focus more on developing new features rather than worrying about
↪  infrastructure management.

### Storytelling Hooks

---

#### Dramatic Question
```

```
Could making a computer dumber actually make it smarter? In this case, by
↪  centralizing management with Kubernetes clusters, you're not reducing the
↪  intelligence of your systems; instead, you're providing a smarter way to
↪  manage and scale them.

#### Point of View
From the perspective of an engineer facing the challenge of managing complex
↪  microservices applications across multiple environments, the story reveals
↪  how Kubernetes clusters offer a powerful solution that simplifies deployment,
↪  scaling, and management.

### Classroom Delivery Tips

---

#### Pacing
- Pause after explaining each key point to ensure students understand before
↪  moving on.
- Ask questions like: "Can you think of a situation where having a cluster would
↪  be beneficial?"
- Summarize the impact at the end to reinforce its importance.

#### Analogy
Imagine you have a team of workers building houses. Each worker is like a
↪  container, and they need to be managed effectively to ensure all houses are
↪  built correctly and on time. Clusters in Kubernetes act like a project
↪  manager who coordinates these workers efficiently, ensuring that resources
↪  are used optimally and the workload can scale up or down based on demand.
↪  This analogy helps students grasp how clusters streamline the management of
↪  containerized applications.

### Interactive Activities for Clusters
### 1. Debate Topic

**Proposition:** "Clusters should be the default deployment method for all
↪  containerized applications due to their unparalleled efficiency in managing
↪  large-scale workloads."

**Opposition:** "Clusters are not always the best solution, as they come with
↪  significant overhead and complexity that can outweigh their benefits for
↪  smaller or less demanding applications."

### 2. What If Scenario Question
```

```
**Scenario:**
Imagine you are a DevOps engineer tasked with setting up a new containerized
↪  application environment for your organization. Your company has recently
↪  decided to migrate from traditional virtual machines (VMs) to Kubernetes
↪  clusters due to the promise of improved efficiency and scalability.

The project manager has given you two options:
- **Option A:** Deploy all applications using a Kubernetes cluster, leveraging
↪  its advanced features like automatic scaling, load balancing, and resource
↪  management.
- **Option B:** Continue with VM-based deployments for simplicity and ease of
↪  use, while still benefiting from container technology where necessary.

**Question:**
Given the strengths and weaknesses (or in this case, the absence of weaknesses)
↪  of Kubernetes clusters as described, which option would you choose? Justify
↪  your decision by considering factors such as application scale, resource
↪  requirements, team expertise, and long-term maintenance costs.


---

## Teaching Module: Master Components
### The Story

#### The Problem (Event)
Imagine you are managing a vast city with millions of people, vehicles, and
↪  buildings. Every day, resources need to be allocated efficiently to ensure
↪  everyone's needs are met. Without any central planning or coordination, chaos
↪  would reign-people might not get the services they need, resources could be
↪  wasted, and overall quality of life would suffer.

In the world of Kubernetes clusters, managing such a large number of containers
↪  is no different. Before the introduction of Master components, each container
↪  or pod was managed individually without any overarching strategy. This led to
↪  inefficient resource utilization, inconsistent application performance, and
↪  high maintenance costs for operators.

#### The 'Aha!' Moment (Experience)
```

```
One day, a brilliant engineer at Google had an "aha!" moment. They realized that
↪ instead of treating every container as a separate entity, they could
↪ centralize the management processes. Thus, the concept of Master components
↪ was born! These master nodes act like the city's central control center,
↪ overseeing everything from scheduling and scaling to health checks.

The Master node has several key components:
- **API Server**: Acts as the front door for all interactions with the cluster.
- **Scheduler**: Decides which container should run on which node based on
↪ resource availability and other factors.
- **Controller Manager**: Monitors and maintains the desired state of the
↪ cluster, ensuring everything is running smoothly.

Together, these components ensure that the cluster operates at peak efficiency,
↪ maintaining a consistent and reliable environment for applications to run.

#### The Impact (Meaning)
The introduction of Master components has transformed how Kubernetes clusters are
↪ managed. They provide centralized control over the entire system, enabling
↪ operators to make informed decisions about resource allocation, application
↪ scaling, and health management. This centralization brings several
↪ advantages:
- **Efficiency**: Resources are used more effectively as the scheduler makes
↪ optimal placement decisions.
- **Reliability**: Applications run smoothly with minimal downtime due to
↪ automated monitoring and recovery mechanisms.

However, it's worth noting that while Master components offer significant
↪ benefits, they also come with potential drawbacks. For example, a single
↪ point of failure could disrupt the entire cluster if not properly managed.
↪ Nonetheless, their strengths far outweigh these weaknesses, making them
↪ indispensable for orchestrating container operations in Kubernetes clusters.

### Storytelling Hooks

#### Dramatic Question
Could making a computer dumber actually make it smarter? By centralizing control
↪ and decision-making processes, we can manage complex systems more
↪ efficiently.

#### Point of View
From the perspective of an engineer facing a challenge with resource allocation
↪ and application performance in a large-scale Kubernetes cluster.
```

```
### Classroom Delivery Tips

- **Pacing**: Pause after explaining each component to ensure students understand
↪  their roles.
  - "Let's take a moment to absorb what the API Server does. How is it like the
  ↪  front door for all interactions?"

- **Analogy**:
  - "Imagine you are organizing a huge party in your city. You can either have
  ↪  everyone making individual decisions about where to put tables and chairs,
  ↪  or you can have one central planner who decides everything. The Master
  ↪  components act like that central planner, ensuring the entire event runs
  ↪  smoothly."

By weaving these elements together, teachers can create an engaging narrative
↪  that helps students grasp the importance of Master components in Kubernetes
↪  clusters.

### Interactive Activities for Master Components
### 1. Debate Topic

**Topic:**
"Is the centralized control provided by Master Components a net positive or
↪  negative for cluster management?"

**Arguments For Centralized Control:**
- Enables consistent policies and processes across the entire cluster.
- Simplifies decision-making and reduces complexity.
- Facilitates easier monitoring, maintenance, and security measures.

**Counterarguments Against Centralized Control:**
- Potential bottleneck if the central point of control fails or experiences high
↪  latency.
- May limit flexibility in local adaptations to specific environments or
↪  workloads.

### 2. What If Scenario Question

**Scenario:**
```

```
Imagine your class is tasked with setting up a new distributed system for
↪  managing a school's library resources, which includes tracking book
↪  checkouts, inventory management, and user access permissions. The system
↪  needs to be scalable, secure, and maintainable by a team of students.

**Question:**
"Considering the strengths and weaknesses of Master Components, should your group
↪  adopt a centralized control approach or opt for a distributed model? Justify
↪  your choice based on potential trade-offs in terms of flexibility, security,
↪  and ease of management."

This scenario forces students to consider the practical implications of using
↪  Master Components in a real-world context, applying their understanding of
↪  the strengths and weaknesses to make an informed decision.


---

## Teaching Module: Kubelets
### The Story (Problem -> Solution -> Impact)

#### The Problem (Event)
Imagine a vast network of computers working together in harmony, like an
↪  orchestra playing a symphony. Each instrument (or node) needs to be perfectly
↪  tuned and ready at any moment to perform its part. But how do you ensure that
↪  every instrument is ready, playing the right notes, and reporting back about
↪  its performance? This is where the challenge lies-especially when there are
↪  countless nodes scattered across different locations.

#### The 'Aha!' Moment (Experience)
Enter Kubelets, the unsung heroes of this digital orchestra. Picture these agents
↪  as the maestros of each node in our Kubernetes cluster. They receive
↪  instructions from the master components and ensure that the containers
↪  running on their respective nodes are functioning correctly. Think of it like
↪  a conductor who checks each musician's instrument to make sure it’s tuned,
↪  listens for any issues, and ensures everyone is playing the right notes.

- **Kubelets communicate with the Master components to receive instructions.**
↪  Just as the maestro gets his score from the composer, Kubelets get their
↪  tasks from the Kubernetes controller.
- **They manage the lifecycle of containers on their respective nodes.** Kubelets
↪  can start up new containers when needed, just like a conductor might call for
↪  more instruments to join in.
```

```
- **Kubelets ensure that containers are running as expected and report back to
↪  the Master.** They keep an eye on everything, ensuring nothing goes awry, and
↪  they provide regular reports so the maestro (the Kubernetes controller) knows
↪  what's happening.

#### The Impact (Meaning)
Kubelets play a critical role in executing container orchestration tasks at the
↪  node level, ensuring that applications run smoothly across the cluster. This
↪  decentralized management allows for efficient scaling and resource
↪  allocation-like having a flexible seating arrangement where instruments can
↪  move around to create different harmonies as needed. By enabling this kind of
↪  adaptive and resilient system, Kubelets make sure our digital symphony is
↪  always in tune.

### Storytelling Hooks

#### Dramatic Question
Could making a computer dumber actually make it smarter? How does an agent that
↪  appears to have less intelligence enable the orchestration of complex tasks
↪  across multiple nodes?

#### Point of View
From the perspective of an engineer facing the challenge of managing a
↪  large-scale distributed system, how would they feel about having these
↪  intelligent agents working behind the scenes?

### Classroom Delivery Tips

- **Pacing**: Start by painting the picture of a large network needing
↪  coordination. Pause here to ask students if they can think of any examples
↪  where coordination is crucial.
- **Analogy**: After introducing Kubelets as maestros, pause and ask, "Can you
↪  imagine how hard it would be to manage an orchestra without a conductor? How
↪  does having a conductor make the process easier?"
- **Relatable Analogy**: Compare Kubelets to housekeepers in a large hotel. Each
↪  housekeeper is responsible for their floor (node), making sure everything is
↪  clean and tidy, reporting any issues, and ensuring guests have what they
↪  need. This can help students understand that just like housekeepers keep the
↪  rooms ready, Kubelets maintain the containers.
- **Engagement**: Encourage students to think about how they might feel if they
↪  were in charge of managing a system with hundreds or thousands of nodes
↪  without tools like Kubelets.
```

```
### Interactive Activities for Kubelets
### 1. Debate Topic

**Statement for Debate:**
"Kubelets are indispensable in modern container orchestration due to their
↪  unparalleled efficiency in decentralized management of containers."

**Teams:**
- **Affirmative Team:** Argues that Kubelets' strengths, particularly their
↪  ability to enable efficient scaling and resource allocation, make them
↪  essential tools in contemporary cloud computing environments.
- **Opposition Team:** Challenges the necessity by questioning if there are
↪  alternative methods or technologies that could achieve similar results
↪  without the specific benefits of Kubelets.

### 2. What If Scenario Question

**Scenario:**
Imagine your class is participating in a hypothetical hackathon where teams must
↪  design and deploy a scalable microservices application using Kubernetes. Your
↪  team has decided to utilize Kubelets for container management. However, you
↪  notice that another group is skeptical about the necessity of Kubelets given
↪  their concerns over potential complexity.

**Question:**
Given the scenario above, how would you justify the use of Kubelets in your
↪  project? Consider discussing at least two key strengths of Kubelets and
↪  explain why these benefits outweigh any perceived complexities or challenges.
```

6

Discussion

This chapter This study aims to explore the potential application of SLMs in educational data narratives. The experimental results demonstrate the feasibility of the system design. During the experiment, we found the following points worthy of discussion.

6.1 Limitation of SLMs

We proposed a scalable pipeline that decomposes the storytelling process into manageable, semantically coherent tasks. This allowed even small models with limited context windows to generate rich, pedagogically aligned content. Despite the promising results, several limitations of SLMs were identified, particularly in how they respond to varying levels of prompt structure.

During zero-shot prompting with informal or unstructured queries, all evaluated SLMs failed to produce uniformly structured outputs. Responses were often vague, disorganized, or incomplete, lacking the expected JSON format. As shown below, even when the intent was to generate a structured response in JSON, the model failed to follow formatting conventions:

```
{
    Q19: Help me develop a lesson on cloud standards and compliance, including
    ↪  NIST guidelines, ISO standards, CSA STAR certifications, and the
    ↪  importance of interoperability and secure multi-cloud operations.
    Raw answer: {
        "question": {
                "topic": "Cloud Standards & Compliance",
                "summary": "This lesson will cover NIST guidelines, ISO
                ↪  standards, CSA STAR certifications, and the importance of
                ↪  interoperability and secure multi-cloud operations."
```

```
    }

[JSON Parser error] Expecting ',' delimiter: line 5 column 2 (char 227)
Attempting fallback with ast.literal_eval...
[AST Parser fallback error] '{' was never closed (<unknown>, line 1)
```

This response is incomplete and fails basic JSON syntax validation due to a missing closing brace. The resulting error, as shown below, reflects an inability to produce even syntactically valid output. Such formatting failures illustrate that SLMs exhibit poor generalization when prompts lack explicit instructional scaffolding.

Only after introducing highly structured zero-shot prompts—designed to include narrative templates, input fields (e.g., Strengths, Weaknesses), and clear formatting rules—did most models demonstrate reliable behavior, as the output from Phi4_14b shown in 5.1. In the new zero-shot settings, output stability varied significantly across models. Olmo2_7b, for instance, showed residual instability: occasional lapses into inconsistent formatting or ignoring instruction tokens altogether, as shown below:

```
{
    "Question": "Prepare a class on modern containerization tools, comparing
    ↪  Docker, Singularity, and Linux Containers, focusing on their unique
    ↪  features, HPC scenarios, and differences from traditional virtualization
    ↪  methods.",
    "Knowledge_Topic": "Containerization",
    "Core_Concepts": [
    {
        "Concept": "Docker",
        "Definition": "An open-source platform that automates the deployment,
        ↪  scaling, and management of applications,"

[Parser error] Got invalid return object.
For troubleshooting, visit:
https://python.langchain.com/docs/troubleshooting/errors/OUTPUT_PARSING_FAILURE
Error: '{' was never closed (<unknown>, line 5)
```

The output is structurally flawed: the "Key_Points", "Significance_Detail", etc, fields were expected to be present. This is also one of the reasons why Olmo underperforms in knowledge extraction tasks.

In contrast to larger LLMs, which often exhibit robust adherence to instruction even with minimal prompting, SLMs remain brittle and require heavy prompt tuning to maintain alignment with task objectives.

6.2 Gap between SLMs and LLMs

To better understand the performance gap between Small Language Models (SLMs) and Large Language Models (LLMs), we conducted a targeted comparative study using a selected subset of the dataset. Specifically, ten questions were randomly selected from the QA dataset introduced in Section 4.4.

For the Knowledge Extraction stage, we used NotebookLM as the LLM-based baseline. NotebookLM is a Google experimental tool designed for grounded, document-based question answering. It offers high transparency, document alignment, and supports structured outputs, making it well-suited for knowledge-intensive educational tasks. Compared to general-purpose models, NotebookLM is optimized for reading multi-source documents and extracting concept-level summaries with high factual consistency.

For the Story Generation stage, we used GPT-4o to generate stories and classroom activities. ChatGPT-4o was selected due to its superior instruction-following ability, narrative fluency, and support for markdown-based output formatting. It also showed strong performance in educational content generation in previous studies.

All outputs were evaluated using the same metrics described in Section 4.5. These included knowledge extraction evaluation and story generation evaluation. The evaluation was conducted using DeepSeek-V3, the same automatic scorer used for SLMs, to ensure consistency and comparability.

The Knowledge Extraction evaluation result is shown in Table 6.1. Notebooklm (3.88) performs better than Phi4_14b (3.67) in overall performance, mainly because of its advantage in the generation stage, especially in answer completeness. Their retrieval scores are almost the same (3.30 vs. 3.27). In the generation stage, its average score (4.27) is 0.33 points higher than Phi4_14b (3.94). It is also more accurate (+0.17) and far more complete (+0.76), though both models perform similarly well in faithfulness (+0.06). Overall, Phi4_14b can still be considered to have capabilities close to larger LLMs when processing text data, especially in retrieval and faithfulness, even though its generation quality is slightly weaker.

Model	Retrieval	Gen Avg	Faithfulness	Accuracy	Completeness	Overall
Notebooklm	3.30	4.27	4.5	4.07	4.23	3.88

Table 6.1: Evaluation of Knowledge Extraction by Notebooklm

The Story Generation evaluation result is shown in Table 6.2. The overall weighted scores of the two systems are very close, with ChatGPT-4o scoring 3.86 and the Phi4_14b + Qwen2.5_7b combination scoring 3.83. However, ChatGPT-4o clearly leads in story generation, with a story score of 3.82 compared to 3.74. It shows large advantages in fluency and consistency (4.0 vs. 3.57) and narrative coherence (3.7 vs. 3.48), making its stories more readable and engaging. In contrast, the Phi4_14b + Qwen2.5_7b combination performs better in factual accuracy (4.07 vs. 3.8) and narrative grounding for activity design (3.48 vs. 3.40), showing it is more reliable in knowledge-related tasks.

The activity scores from both models are comparable, with the Phi4_14b + Qwen2.5_7b (3.48) scoring slightly higher in narrative grounding, indicating better integration of activities within the story context. Meanwhile, the GPT-4o (4.40) achieves a higher classroom applicability score then Phi4_14b + Qwen2.5_7b (4.35), reflecting stronger practical usability in educational settings. This suggests the SLM combination at contextual relevance, while the GPT-4o better supports classroom implementation.

These differences indicate that ChatGPT-4o is more suitable when narrative quality is critical, while the Phi4_14b + Qwen2.5_7b combination demonstrates that SLMs can be a feasible choice in scenarios that require strict factual accuracy, even if narrative quality is partially sacrificed.

Model	Structure	Factual Accuracy	Narrative Coherence	Educational Engagement	Fluency & Consistency	Narrative Grounding	Classroom Applicability	Overall
ChatGPT-4o	3.50	3.80	3.70	4.10	4.00	3.41	4.40	3.86

Table 6.2: Evaluation of Story Generation by ChatGPT-4o

6.3 Generalization and Guidelines

To help educators and researchers apply the methodology introduced in this thesis, this appendix presents a step-by-step guide to building educational data storytelling systems using RAG and SLMs. This guide supports generalization across subjects and educational contexts.

6.3.1 Overview of the Pipeline

The proposed pipeline consists of three modular components:

1. **Knowledge Extraction** using hybrid retrieval and chunking techniques.

2. **Story Generation** based on structured prompts and narrative scaffolds.

3. **Interactive Narration** through classroom activities and discussion design.

Each component can be adapted independently to different curricula, domains, and language models.

6.3.2 Step-by-Step Implementation Guide

1. **Data Preparation**

   - Gather source materials (e.g., textbooks, PDFs)
   - Parse documents using PDF parsers (e.g., `minerU`) to retain structure
   - Apply a two-stage chunking process:
     - *Semantic Chunking*: split by conceptual boundaries
     - *Recursive Chunking*: further broken down into smaller, hierarchical chunks
   - Annotate each chunk with metadata (ID, type, topic)

2. **RAG Setup**

   - Implement a hybrid retrieval system:
     - **Sparse**: BM25 for exact matches
     - **Dense**: ChromaDB with HuggingFace embeddings for semantic search
   - Use a cross-encoder (e.g., BAAI/bge-reranker-v2-m3) to re-rank retrieved chunks
   - Store top-k contextually relevant chunks for generation
   - Output the extraction results in a structured format

3. **Selecting Appropriate SLMs**

   Use different models for each stage in a modular pipeline

   - **Knowledge Extraction:** Phi4_14b or Qwen2.5_7b (for factual accuracy)
   - **Story Generation:** Qwen2.5_7b or Llama3.1_8b (for narrative quality)

4. **Prompt Engineering Strategy**

   - Pass the Knowledge Extraction Results
   - Use modular and chained prompts

- Structure story prompts using *Problem* → *Solution* → *Impact*, embed hooks such as analogies, questions, and character perspectives
- Create classroom activities from concept strengths/weaknesses

5. **Customization**

- Teachers can review, edit, or regenerate each module
- Export outputs in Markdown, PDF formats

6.3.3 Summary Checklist

Step	Task	Tool/Model
1	Data Parsing & Chunking	minerU, semantic + token-based splitter
2	RAG Setup	BM25, ChromaDB, Cross-Encoder
3	SLM Selection	Phi4_14b, Qwen2.5_7b
4	Prompt Engineering	Modular, role-based, CoT prompts

Table 6.3: Checklist for Building an Educational Data Storytelling System

6.4 Future Directions

This research provides a foundation for using SLMs in educational content generation. However, several directions can be explored to improve the system's capabilities and broaden its application in future work.

6.4.1 Integration of Multimodal SLMs

The current system only supports text-based inputs and outputs. Educational materials such as textbooks, slides, and handouts often contain diagrams, tables, and other visual elements. In the present implementation, PDF files must be converted into Markdown format through preprocessing. This conversion process is computationally expensive and may lose layout or contextual information during parsing when facing complex inputs.

Future versions of the system can integrate multimodal SLMs that accept both text and visual data. These models can directly process images, scanned documents, and charts without conversion. This would reduce the need for manual formatting and improve the system's ability to extract information from diverse sources. Multimodal support can also enhance storytelling by generating content that includes both narrative and visual aids, such as timelines, concept maps, or annotated diagrams.

6.4.2 Platform Development

At present, the system has only been tested in a controlled research environment. It operates through scripts and local interfaces, which are not accessible to general users such as teachers or curriculum designers. For future development, building a user-friendly platform is essential to support real-world classroom integration.

A web-based platform can provide graphical interfaces for uploading course materials, reviewing generated stories, and editing activities. This would allow non-technical users to operate the system without needing knowledge of prompts or model orchestration. In addition, user authentication, session history, and cloud-based model serving could be added to support long-term use and scalability. These enhancements would transform the current prototype into a practical educational tool suitable for daily classroom use.

7

Conclusion

This thesis explored the potential of Small Language Models as an efficient and practical tool for educational data storytelling. By dividing the process into knowledge extraction and story generation, the system allowed different models to focus on tasks that matched their strengths. The knowledge extraction module ensured factual accuracy and relevance, while the story generation module produced coherent and engaging narratives through a chained-prompting strategy. An extensive evaluation of 49 model combinations demonstrated that SLMs can generate classroom-ready materials with high accuracy and strong narrative quality, confirming their feasibility for educational applications.

The system's performance was further validated through comparisons with larger language models. Although large models still have advantages in narrative fluency and stylistic consistency, the optimized SLM pipeline achieved competitive results, particularly in factual reliability and contextual grounding. These findings indicate that SLMs can be a cost-effective and accessible alternative for educational settings, even when some creative flexibility is sacrificed. The modular architecture proved essential for overcoming the inherent limitations of smaller models, highlighting the importance of strategic system design.

Future work will focus on integrating multimodal SLMs and developing user-friendly tools to increase adoption among educators. With further optimization, SLM-based systems can help the creation of AI-generated educational materials, reduce barriers to high-quality content production, and enrich the learning experience for students. This research shows that, with reasonable design, SLMs are not just scaled-down versions of larger models. They can be a powerful and practical solution for bringing generative AI into education.

References

[1] Craig Eilert Abrahamson. **Storytelling as a pedagogical tool in higher education**. *Education*, **118**(3):440–452, 1998. 1, 18

[2] Halah Ahmed Alismail. **Integrate digital storytelling in education.** *Journal of Education and Practice*, **6**(9):126–129, 2015. 1, 18

[3] Shen Wang, Tianlong Xu, Hang Li, Chaoli Zhang, Joleen Liang, Jiliang Tang, Philip S Yu, and Qingsong Wen. **Large language models for education: A survey and outlook**. *arXiv preprint arXiv:2403.18105*, 2024. 1, 13

[4] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. **Retrieval-augmented generation for knowledge-intensive nlp tasks**. *Advances in neural information processing systems*, **33**:9459–9474, 2020. 5

[5] Yue Zhang, Yafu Li, Leyang Cui, Deng Cai, Lemao Liu, Tingchen Fu, Xinting Huang, Enbo Zhao, Yu Zhang, Yulong Chen, et al. **Siren's song in the AI ocean: a survey on hallucination in large language models**. *arXiv preprint arXiv:2309.01219*, 2023. 5

[6] Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yixin Dai, Jiawei Sun, Haofen Wang, and Haofen Wang. **Retrieval-augmented generation for large language models: A survey**. *arXiv preprint arXiv:2312.10997*, **2**(1), 2023. 5

[7] Luyu Gao, Xueguang Ma, Jimmy Lin, and Jamie Callan. **Precise zero-shot dense retrieval without relevance labels**. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1762–1777, 2023. 5

[8] Xinbei Ma, Yeyun Gong, Pengcheng He, Hai Zhao, and Nan Duan. **Query rewriting in retrieval-augmented large language models**. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 5303–5315, 2023. 5, 6

[9] Vladimir Blagojevi. **Enhancing rag pipelines in haystack: Introducing diversityranker and lostinthemiddleranker**, 2023. 5

[10] Yunfan Gao, Yun Xiong, Meng Wang, and Haofen Wang. **Modular rag: Transforming rag systems into lego-like reconfigurable frameworks**. *arXiv preprint arXiv:2407.21059*, 2024. 6

[11] Xue Wu and Kostas Tsioutsiouliklis. **Thinking with Knowledge Graphs: Enhancing LLM Reasoning Through Structured Data**. *arXiv preprint arXiv:2412.10654*, 2024. 6

[12] Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. **Interleaving retrieval with chain-of-thought reasoning for knowledge-intensive multi-step questions**. *arXiv preprint arXiv:2212.10509*, 2022. 6

[13] Jerry Liu. **LlamaIndex**, 11 2022. 6

[14] Xianming Li and Jing Li. **AnglE-optimized Text Embeddings**. *arXiv preprint arXiv:2309.12871*, 2023. 6

[15] Shitao Xiao, Zheng Liu, Peitian Zhang, and Niklas Muennighoff. **C-Pack: Packaged Resources To Advance General Chinese Embedding**, 2023. 6

[16] Yunjia Xi, Jianghao Lin, Weiwen Liu, Xinyi Dai, Weinan Zhang, Rui Zhang, Ruiming Tang, and Yong Yu. **A bird's-eye view of reranking: from list level to page level**. In *Proceedings of the Sixteenth ACM International Conference on Web Search and Data Mining*, pages 1075–1083, 2023. 6

[17] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. **Language models are unsupervised multitask learners**. *OpenAI blog*, **1**(8):9, 2019. 6

[18] Pranab Sahoo, Ayush Kumar Singh, Sriparna Saha, Vinija Jain, Samrat Mondal, and Aman Chadha. **A systematic survey of prompt engineering in large language models: Techniques and applications**. *arXiv preprint arXiv:2402.07927*, 2024. 6

[19] David Van Buren. **Guided scenarios with simulated expert personae: a remarkable strategy to perform cognitive work**. *arXiv preprint arXiv:2306.03104*, 2023. 7

[20] Banghao Chen, Zhaofeng Zhang, Nicolas Langrené, and Shengxin Zhu. **Unleashing the potential of prompt engineering for large language models**. *Patterns*, 2025. 7

[21] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. **Chain-of-thought prompting elicits reasoning in large language models**. *Advances in neural information processing systems*, **35**:24824–24837, 2022. 7

[22] Kunlun Zhu, Yifan Luo, Dingling Xu, Yukun Yan, Zhenghao Liu, Shi Yu, Ruobing Wang, Shuo Wang, Yishan Li, Nan Zhang, et al. **Rageval: Scenario specific rag evaluation dataset generation framework**. *arXiv preprint arXiv:2408.01262*, 2024. 7

[23] Hao Yu, Aoran Gan, Kai Zhang, Shiwei Tong, Qi Liu, and Zhaofeng Liu. **Evaluation of retrieval-augmented generation: A survey**. In *CCF Conference on Big Data*, pages 102–120. Springer, 2024. 7

[24] Yao Fu, Hao Peng, Litu Ou, Ashish Sabharwal, and Tushar Khot. **Specializing smaller language models towards multi-step reasoning**. In *International Conference on Machine Learning*, pages 10421–10430. PMLR, 2023. 7

[25] Elias Frantar and Dan Alistarh. **Sparsegpt: Massive language models can be accurately pruned in one-shot**. In *International Conference on Machine Learning*, pages 10323–10337. PMLR, 2023. 7

[26] Jing Liu, Ruihao Gong, Xiuying Wei, Zhiwei Dong, Jianfei Cai, and Bohan Zhuang. **Qllm: Accurate and efficient low-bitwidth quantization for large language models**. *arXiv preprint arXiv:2310.08041*, 2023. 7

[27] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. **DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter**. *arXiv preprint arXiv:1910.01108*, 2019. 7

[28] Zhiqing Sun, Hongkun Yu, Xiaodan Song, Renjie Liu, Yiming Yang, and Denny Zhou. **Mobilebert: a compact task-agnostic bert for resource-limited devices**. *arXiv preprint arXiv:2004.02984*, 2020. 7

[29] Xiao Bi, Deli Chen, Guanting Chen, Shanhuang Chen, Damai Dai, Chengqi Deng, Honghui Ding, Kai Dong, Qiushi Du, Zhe Fu, et al. **Deepseek llm: Scaling open-source language models with longtermism**. *arXiv preprint arXiv:2401.02954*, 2024. 8

[30] Gemma Team, Thomas Mesnard, Cassidy Hardin, Robert Dadashi, Surya Bhupatiraju, Shreya Pathak, Laurent Sifre, Morgane Rivière, Mihir Sanjay Kale, Juliette Love, et al. **Gemma: Open models based on gemini research and technology**. *arXiv preprint arXiv:2403.08295*, 2024. 8

[31] Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. **The llama 3 herd of models**. *arXiv preprint arXiv:2407.21783*, 2024. 8

[32] Team OLMo, Pete Walsh, Luca Soldaini, Dirk Groeneveld, Kyle Lo, Shane Arora, Akshita Bhagia, Yuling Gu, Shengyi Huang, Matt Jordan, Nathan Lambert, Dustin Schwenk, Oyvind Tafjord, Taira Anderson, David Atkinson, Faeze Brahman, Christopher Clark, Pradeep Dasigi, Nouha Dziri, Michal Guerquin, Hamish Ivison, Pang Wei Koh, Jiacheng Liu, Saumya Malik, William Merrill, Lester James V. Miranda, Jacob Morrison, Tyler Murray, Crystal Nam, Valentina Pyatkin, Aman Rangapur, Michael Schmitz, Sam Skjonsberg, David Wadden, Christopher Wilhelm, Michael Wilson, Luke Zettlemoyer, Ali Farhadi, Noah A. Smith, and Hannaneh Hajishirzi. **2 OLMo 2 Furious**, 2024. 8

[33] Guan Wang, Sijie Cheng, Xianyuan Zhan, Xiangang Li, Sen Song, and Yang Liu. **OpenChat: Advancing Open-source Language Models with Mixed-Quality Data**. *arXiv preprint arXiv:2309.11235*, 2023. 8

[34] Qwen Team. **Qwen2.5: A Party of Foundation Models**, September 2024. 8

[35] Marah Abdin, Jyoti Aneja, Harkirat Behl, Sébastien Bubeck, Ronen Eldan, Suriya Gunasekar, Michael Harrison, Russell J Hewett, Mojan Javaheripi, Piero Kauffmann, et al. **Phi-4 technical report**. *arXiv preprint arXiv:2412.08905*, 2024. 8

[36] Adam S.Z. Belloum Junming Ye. **Data Storytelling: A Systematic Review Across Domains**, 08 2025. 8

[37] A Shaji George. **The potential of generative AI to reform graduate education**. *Partners Universal International Research Journal*, **2**(4):36–50, 2023. 11

[38] Dominik Thüs, Sarah Malone, and Roland Brünken. **Exploring generative AI in higher education: a RAG system to enhance student engagement with scientific literature**. *Frontiers in Psychology*, **15**:1474892, 2024. 11

[39] Eason Chen, Danyang Wang, Luyi Xu, Chen Cao, Xiao Fang, and Jionghao Lin. **A Systematic Review on Prompt Engineering in Large Language Models for K-12 STEM Education**. *arXiv preprint arXiv:2410.11123*, 2024. 12

[40] Hang Jiang, Xiajie Zhang, Robert Mahari, Daniel Kessler, Eric Ma, Tal August, Irene Li, Alex'Sandy' Pentland, Yoon Kim, Deb Roy, et al. **Leveraging large language models for learning complex legal concepts through storytelling**. *arXiv preprint arXiv:2402.17019*, 2024. 12

[41] Ronen Eldan and Yuanzhi Li. **Tinystories: How small can language models be and still speak coherent english?** *arXiv preprint arXiv:2305.07759*, 2023. 12

[42] Mostafa Faghih Shojaei, Rahul Gulati, Benjamin A Jasperson, Shangshang Wang, Simone Cimolato, Dangli Cao, Willie Neiswanger, and Krishna Garikipati. **AI-University: An LLM-based platform for instructional alignment to scientific classrooms**. *arXiv preprint arXiv:2504.08846*, 2025. 13

[43] Zhijun Pan, Antonios Andronis, Eva Hayek, Oscar AP Wilkinson, Ilya Lasy, Annette Parry, Guy Gadney, Tim J Smith, and Mick Grierson. **Guiding Generative Storytelling with Knowledge Graphs**. *arXiv preprint arXiv:2505.24803*, 2025. 13

Appendix A

Examples of Story Generation with Model Combinations

This appendix presents detailed examples of story generation experiments using different model combinations. These examples are drawn from both the competitive middle-tier and underperforming groups described in the main results section. They are intended to illustrate the qualitative differences in output across tiers.

A.1 Competitive Middle-Tier

Story Generation from Phi4_14b + Openchat_7b (score: 3.50)

```
 ```markdown
# Lesson Title: Kubernetes: Orchestrating Microservices with Containers

1. **Introduction (Hook)**: Discover how Kubernetes revolutionizes
↪  container-based microservice architectures by tackling a real-world scenario
↪  where efficient scaling and resource management are crucial.

2. **Core Content Delivery**:
   1. Pods: Understand the role of Pods as the smallest deployable units in
   ↪  Kubernetes, housing one or more containers that share resources and
   ↪  network connectivity.
   2. Clusters: Learn how Clusters consist of multiple nodes running
   ↪  containerized applications, managed by a Master component to ensure high
   ↪  availability and fault tolerance.
   3. Master Components: Explore the key Master components such as the API
   ↪  server, etcd, controller manager, and scheduler that maintain the desired
   ↪  state of the cluster and manage Pods.
```
```

```
   4. Kubelets: Investigate how each node in a Cluster runs a kubelet responsible
   ↪  for communicating with the Master components and maintaining container
   ↪  lifecycles.

3. **Key Activity/Discussion**: Participate in a group activity where students
↪  simulate the process of creating a Pod, deploying it to a Cluster, and
↪  managing its lifecycle using Kubernetes commands.

4. **Conclusion & Synthesis**: Reflect on how Kubernetes efficiently orchestrates
↪  containerized microservices through Pods, Clusters, Master components, and
↪  kubelets, enabling scalability and robust management across diverse
↪  environments.
```

Teaching Module: Pods
1. The Story (Problem -> Solution -> Impact)
The Problem (Event):
_Once upon a time in a faraway land called Kubernetes, there was a big problem.
↪ Developers were struggling to manage their applications and services
↪ efficiently. They needed a way to deploy, scale, and manage their containers
↪ without much hassle._

The 'Aha!' Moment (Experience):
_One day, the wise developers stumbled upon the concept of "Pods". A Pod was like
↪ a tiny magical box that could contain one or more containers. These
↪ containers shared resources like networking and storage, which made it easier
↪ for them to work together and cooperate._

_Within this magical box, the containers could communicate with each other as if
↪ they were part of the same team. They were managed by Kubernetes as a single
↪ entity, making the lifecycle and resource management of these containers much
↪ simpler._

The Impact (Meaning):
_The introduction of Pods in Kubernetes changed everything. It made deploying
↪ microservices within a containerized environment a breeze, and it facilitated
↪ efficient scaling and management of those services. This concept became
↪ crucial for managing the lifecycle of containers, and it helped developers to
↪ focus on their core tasks while leaving the heavy lifting to Kubernetes._

```
_Pods were like superheroes that simplified the deployment process by grouping
↪  related containers together. They made it easier to manage their lifecycle
↪  and resources, which in turn empowered developers to create more efficient
↪  applications and services._

### 2. Storytelling Hooks
- **Dramatic Question**: _What if we could create a unified entity that would
↪  simplify the deployment of multiple containers while sharing resources among
↪  them?_
- **Point of View**: _From the perspective of an overworked developer in need of
↪  a more efficient way to manage their microservices._

### 3. Classroom Delivery Tips
- **Pacing**: _Introduce the concept of Pods, pause for a moment to let the
↪  students absorb the information. Then, continue explaining how they work and
↪  their significance. Ask questions along the way to keep the students
↪  engaged._
- **Analogy**: _Think of a Pod as a small, efficient team of superheroes working
↪  together, each with its own unique powers, but sharing resources like a
↪  common base of operations. They are managed by Kubernetes, which is their
↪  wise and powerful leader that ensures they work efficiently and in harmony._

### Interactive Activities for Pods
 1. Debate Topic: "While Pods simplify the deployment process by grouping related
 ↪  containers together for easier management of lifecycle and resources, they
 ↪  may also limit flexibility in certain scenarios. Is it beneficial to
 ↪  prioritize ease of management over potential limitations in resource
 ↪  allocation?"

2. What If Scenario Question: "Imagine you are a developer tasked with deploying
↪  a large-scale application that requires the efficient use of resources across
↪  multiple containers. If you were to use Pods, how would this affect your
↪  ability to optimize resource usage, and what alternative approaches might be
↪  considered?"


---

## Teaching Module: Clusters
 ### 1. The Story (Problem -> Solution -> Impact)
#### The Problem (Event):
```

```
Once upon a time in a bustling city of cloud-native applications, there was a
↪  growing need for efficient and flexible infrastructure to run and manage
↪  large-scale containerized workloads. Applications were spread across public,
↪  private, and hybrid clouds, and they all had their unique challenges.

#### The 'Aha!' Moment (Experience):
One day, a group of brilliant engineers discovered a magical solution -
↪  Kubernetes Clusters! These clusters were like a team of powerful superheroes
↪  working together to run containerized applications. They were made up of
↪  nodes that collaborated and shared resources, providing the necessary
↪  infrastructure for running containers at scale. The teams could quickly scale
↪  their workloads when needed, making it easy to add or remove nodes from the
↪  cluster.

#### The Impact (Meaning):
The introduction of Kubernetes Clusters brought about a revolution in the way
↪  applications were managed. They provided a powerful platform that enabled
↪  cloud-native applications to grow and adapt, ensuring they remained scalable
↪  and flexible across different environments. By supporting both on-premise and
↪  cloud deployments, these clusters became essential for hosting large-scale
↪  containerized workloads efficiently.

### 2. Storytelling Hooks
#### Dramatic Question:
Imagine a world where applications can scale up or down instantly, adapting to
↪  their needs without any hiccups. How would that transform the way we build
↪  and manage software?

#### Point of View:
View the story from the perspective of an engineer struggling with managing
↪  containerized workloads across multiple environments, who stumbles upon
↪  Kubernetes Clusters and discovers a solution that changes everything.

### 3. Classroom Delivery Tips
#### Pacing:
- Introduce the problem faced by engineers and the growing need for efficient
↪  infrastructure to manage large-scale containerized workloads.
- As you explain what clusters are, pause to let students grasp the concept. Ask
↪  a question like, "What do you think is the main advantage of having nodes
↪  working together in a cluster?"
- When discussing the strengths of clusters, ask students to identify which
↪  aspect they find most beneficial.
```

```
#### Analogy:
Think of a Kubernetes Cluster as a team of superheroes, where each node is like a
↪  member of the team. They all have unique abilities and work together to
↪  protect the city (containerized applications) from challenges like growing
↪  demand or shrinking resources. The cluster ensures that no single member is
↪  overwhelmed, and they can all adapt quickly to changing situations.

### Interactive Activities for Clusters
 1. Debate Topic: "Kubernetes Clusters, while enabling efficient management of
 ↪  large-scale containerized workloads, may not be suitable for smaller scale
 ↪  deployments due to its complexity. Is it worth the investment in learning
 ↪  and implementing Kubernetes for small-scale projects?"

2. What If Scenario Question: "What if a company decided to switch from using
↪  individual virtual machines to a Kubernetes cluster for managing their
↪  containerized applications? How would this decision impact the efficiency,
↪  scalability, and cost of their operations, considering the complexity
↪  involved in managing Kubernetes clusters?"


---

## Teaching Module: Master Components
 ### 1. The Story
**The Problem (Event)**: In the world of containerized applications, there was
↪  once a company facing significant challenges in managing their growing fleet
↪  of containers. They struggled to ensure that resources were efficiently
↪  utilized and that their applications remained reliable across their sprawling
↪  infrastructure.

**The 'Aha!' Moment (Experience)**: One day, they discovered Kubernetes, an
↪  open-source platform designed to automate deployment, scaling, and management
↪  of containerized applications. At the heart of this powerful system lay the
↪  Master Components. These components controlled the scheduling, scaling, and
↪  health management of containers within the cluster. The API server acted as
↪  the central control point for all interactions with Kubernetes, while the
↪  Scheduler decided where to place new containers based on available resources
↪  and requirements. Finally, the Controller Manager ensured that the desired
↪  state of the cluster was maintained at all times.
```

```
**The Impact (Meaning)**: With these Master Components in place, the company
↪  could effectively orchestrate their container operations across the entire
↪  Kubernetes cluster. This centralized control allowed them to maintain
↪  consistent management and decision-making processes, leading to more
↪  efficient resource utilization and increased application reliability. Despite
↪  its importance, it's worth noting that over-reliance on a single point of
↪  control could potentially lead to vulnerabilities if not managed carefully.

### 2. Storytelling Hooks
**Dramatic Question**: What if you could manage your entire containerized
↪  infrastructure with just one set of components?

**Point of View**: From the perspective of an engineer tasked with managing a
↪  complex, multi-container environment.

### 3. Classroom Delivery Tips
**Pacing**: Pause after describing the challenges faced by the company to
↪  emphasize the need for a solution like Kubernetes. When explaining the Master
↪  Components, pause after each key point to ensure understanding and ask
↪  questions to keep students engaged.

**Analogy**: Imagine you're running a bakery with many different types of cakes
↪  and pastries. The Master Components are like the head chef who decides which
↪  desserts to bake, where they should be placed in the display case, and
↪  whether more ingredients need to be ordered or not. They ensure that all the
↪  other chefs (containers) have what they need to create delicious treats
↪  without overcrowding the kitchen (cluster).

### Interactive Activities for Master Components
 1. Debate Topic: "Master Components provide centralized control over a cluster,
  ↪  enabling consistent management and decision-making processes, but they may
  ↪  also lead to reliance on a single point of failure. Should organizations
  ↪  prioritize the benefits of Master Components or focus on building redundancy
  ↪  and resilience into their systems?"

2. What If Scenario Question: "Imagine a company has just adopted the Master
↪  Component model for their server cluster, which has significantly improved
↪  their efficiency and decision-making processes. Suddenly, a major malfunction
↪  occurs in the central Master Component. Given this scenario, should the
↪  company invest more resources into repairing the Master Component or focus on
↪  developing a distributed system that can function independently of any single
↪  component?"
```

```
---

## Teaching Module: Kubelets
 ### The Story

#### The Problem (Event)
Once upon a time in a vast, bustling city of computers, there was a growing
↪  challenge. With more and more applications requiring efficient and reliable
↪  execution across multiple nodes, the need to manage these containers became
↪  increasingly complex and difficult to handle.

#### The 'Aha!' Moment (Experience)
As the story goes, one day, an ingenious group of developers stumbled upon a
↪  concept called Kubelets. Kubelets were agents that run on each node in a
↪  Kubernetes cluster. They were responsible for maintaining the desired state
↪  of containers. You see, Kubelets communicate with the Master components to
↪  receive instructions. They manage the lifecycle of containers on their
↪  respective nodes. Most importantly, they ensure that containers are running
↪  as expected and report back to the Master.

#### The Impact (Meaning)
This discovery was a game-changer for the city of computers. Kubelets played a
↪  critical role in executing container orchestration tasks at the node level,
↪  ensuring that applications ran smoothly across the cluster. They enabled
↪  decentralized management of containers, allowing for efficient scaling and
↪  resource allocation. The benefits were clear - improved efficiency, better
↪  reliability, and increased scalability. However, it also came with
↪  trade-offs, as no solution is perfect. But overall, Kubelets made managing
↪  container orchestration a little less daunting and a lot more effective.

### Storytelling Hooks
- **Dramatic Question**: Can you imagine a world where every computer in a
↪  cluster works together seamlessly, like a well-oiled machine?
- **Point of View**: Let's explore this concept from the perspective of an
↪  overworked system administrator trying to maintain order amidst chaos.

### Classroom Delivery Tips
- **Pacing**: Start by introducing the challenge faced in managing containers
↪  across multiple nodes. Pause for a moment and ask students if they can relate
↪  to this problem. Then, introduce the concept of Kubelets and their role.
↪  Pause again after discussing each Key Point to allow students to absorb the
↪  information.
```

```
- **Analogy**: To help students understand the concept of Kubelets, imagine a
↪  team of chefs (nodes) in a busy kitchen (cluster), responsible for preparing
↪  dishes (containers). The head chef (Master component) instructs the team on
↪  what to prepare and how, while each chef reports back on their progress. This
↪  analogy helps to illustrate the communication and coordination involved in
↪  Kubelets' operations.

### Interactive Activities for Kubelets
 1. Debate Topic: "Kubelets enable decentralized management of containers,
 ↪  allowing for efficient scaling and resource allocation; however, some argue
 ↪  that this comes at the cost of increased complexity in container
 ↪  orchestration. In a world where simplicity and ease of use are highly
 ↪  valued, is it worth adopting Kubelets for their management capabilities?"

2. What If Scenario Question: "Imagine you are tasked with managing a large-scale
↪  containerized application that requires high availability and efficient
↪  resource utilization. Your team has been considering using Kubelets to manage
↪  the containers. However, your team is divided between those who believe in
↪  the benefits of decentralized management, scalability, and resource
↪  allocation provided by Kubelets, and those who are concerned about the added
↪  complexity this might introduce. What if a sudden surge in traffic occurs,
↪  and you have to decide whether to use Kubelets or stick with traditional
↪  container management? Justify your choice based on the trade-offs between the
↪  strengths and weaknesses of Kubelets."
```

A.2 Underperformers

Story Generation from Phi4_14b + Phi4_14b (score: 3.28)

```
# Lesson Plan Outline

## 1. **Lesson Title**
- "Mastering Kubernetes: Orchestrating Microservices with Ease"

## 2. Introduction (Hook)
- **Objective:** Engage students by presenting a real-world scenario where
↪  scaling microservice-based architectures efficiently is critical, prompting
↪  them to consider how orchestration can solve such challenges.

## 3. Core Content Delivery
- **Objective:** Sequentially introduce the core concepts of Kubernetes and their
↪  roles in orchestrating containerized applications.

1. **Pods**
```

```
   - Objective: Explain that Pods are the smallest deployable units in
   ↪  Kubernetes, encapsulating one or more containers to ensure they run
   ↪  together.

2. **Clusters**
   - Objective: Describe how Clusters provide a pool of resources, including
   ↪  multiple nodes where Pods can be scheduled and executed.

3. **Master Components**
   - Objective: Outline the role of Master components (now known as Control
   ↪  Plane) in managing the state of the cluster, scheduling workloads, and
   ↪  maintaining desired states.

4. **Kubelets**
   - Objective: Illustrate how Kubelets are agents running on each node that
   ↪  ensure containers within Pods are healthy and operational.

## 4. Key Activity/Discussion
- **Objective:** Conduct an interactive simulation or group discussion where
↪  students apply the concepts to a hypothetical deployment scenario,
↪  identifying which Kubernetes components would be involved in scaling their
↪  solution.

## 5. Conclusion & Synthesis
- **Objective:** Summarize how Pods, Clusters, Master Components, and Kubelets
↪  collaborate within Kubernetes to efficiently scale microservice-based
↪  architectures, reinforcing their understanding of orchestration principles.


---

## Teaching Module: Pods
## 1. The Story (Problem -> Solution -> Impact)

### The Problem (Event)
In a bustling tech company named "TechCrafters," teams were struggling with
↪  deploying and managing applications efficiently. Each application consisted
↪  of multiple components that needed to work together seamlessly, yet each
↪  component was packaged in separate containers. These containers had to share
↪  resources such as storage and networking but faced challenges due to their
↪  isolated nature. Deploying updates became a complex task, often leading to
↪  downtime or inconsistent states between components.
```

```
### The 'Aha!' Moment (Experience)
One day, Emma, an innovative software engineer at TechCrafters, stumbled upon the
↪ concept of "Pods" while exploring Kubernetes-a powerful system for managing
↪ containerized applications. She learned that a Pod is the smallest deployable
↪ unit in Kubernetes capable of containing one or more containers that share
↪ storage and network resources. This discovery was transformative: Pods
↪ allowed these containers to be deployed together as a single entity, managed
↪ by Kubernetes.

Emma realized that with Pods, she could group related containers into cohesive
↪ units, making it easier to ensure they were always synchronized and
↪ efficiently using shared resources. These Pods facilitated the deployment of
↪ microservices within their containerized environment, addressing many of
↪ TechCrafters' deployment challenges.

### The Impact (Meaning)
The adoption of Pods at TechCrafters revolutionized their deployment processes.
↪ With Pods, managing the lifecycle of containers became streamlined, enabling
↪ efficient scaling and management of their microservice-based architectures.
↪ This innovation significantly reduced downtime during updates and improved
↪ resource utilization. While Pods simplified deployments by grouping related
↪ containers, there were no significant weaknesses that overshadowed these
↪ strengths.

Pods became crucial for TechCrafters, allowing them to maintain a robust and
↪ agile application infrastructure, ultimately leading to happier customers and
↪ a more innovative company culture.

## 2. Storytelling Hooks

- **Dramatic Question**: "How can the smallest unit in a containerized world
↪  transform how applications are deployed and managed?"
- **Point of View**: From the perspective of an engineer like Emma, who faces
↪  daily challenges in managing complex application deployments.

## 3. Classroom Delivery Tips

- **Pacing**:
  - Pause after introducing the problem to allow students to reflect on similar
  ↪  experiences they might have encountered with resource sharing or isolated
  ↪  components.
```

```
  - After explaining what a Pod is and how it works, ask students if anyone can
  ↪  think of another scenario where grouping related items could simplify
  ↪  management.

- **Analogy**:
  - Imagine Pods as a team bus in a relay race. Each container is like an
  ↪  individual runner who needs to pass the baton smoothly to the next. The Pod
  ↪  is the bus that carries all runners, ensuring they start together and have
  ↪  access to shared resources (like water stations), making the entire process
  ↪  more efficient. Without the bus, each runner would need their own vehicle,
  ↪  complicating coordination and resource sharing.

This approach helps students visualize how Pods streamline container management
↪  by grouping related components into a single deployable unit, much like
↪  runners on a team bus in a relay race.

### Interactive Activities for Pods
### Debate Topic

**Statement:** "Given that pods simplify deployment by grouping related
↪  containers together and have no notable weaknesses, they are an essential
↪  component of any container orchestration strategy."

**Arguments For:**
- Simplifies the management process for developers by bundling related
↪  components.
- Enhances resource allocation efficiency through shared namespaces.

**Arguments Against:**
- Lacks explicit weakness may overlook potential limitations or challenges that
↪  could arise in specific use cases.
- May lead to over-reliance on a single abstraction, possibly neglecting other
↪  beneficial orchestration tools or methods.

### What If Scenario Question

Imagine you are part of a development team tasked with deploying a complex
↪  microservices-based application. The application consists of multiple
↪  interdependent services that need to communicate seamlessly and share certain
↪  resources like volumes and network configurations. You decide to use
↪  Kubernetes for orchestrating the containers.
```

```
**Scenario:** Your project manager suggests using pods extensively due to their
↪  ability to group related containers together, thereby simplifying deployment
↪  processes.

**Question:**
- How would you justify the decision to rely heavily on pods given their
↪  strengths? Consider any potential indirect challenges that could arise
↪  despite the absence of explicit weaknesses.

**Considerations for Justification:**
- Discuss how grouping services in pods could streamline deployment and resource
↪  management.
- Reflect on any possible limitations or unforeseen issues, such as scalability
↪  concerns or complexities in managing inter-pod communication.

This scenario encourages students to think critically about the implications of
↪  using pods extensively and explore potential trade-offs that may not be
↪  immediately apparent.


---

## Teaching Module: Clusters
## 1. The Story (Problem -> Solution -> Impact)

### The Problem (Event)
In the fast-paced world of technology, businesses are constantly seeking ways to
↪  enhance their application deployment and management capabilities. Before the
↪  advent of Kubernetes clusters, companies faced significant challenges in
↪  scaling applications efficiently across diverse environments. Managing
↪  containerized applications manually was cumbersome, error-prone, and limited
↪  scalability. Businesses required a solution that could offer both flexibility
↪  and reliability for deploying applications at scale.

### The 'Aha!' Moment (Experience)
One day, an innovative group of engineers encountered this daunting challenge:
↪  how to streamline the management of numerous containers across different
↪  environments effectively? Their breakthrough came when they discovered
↪  Kubernetes clusters-a concept that fundamentally changed their approach. A
↪  Cluster in Kubernetes is a group of nodes that work together to run
↪  containerized applications.
```

```
These clusters can span public, private, or hybrid clouds, offering unparalleled
↪ flexibility. They provide the necessary infrastructure for running and
↪ managing containers at scale, facilitating rapid scaling and workload
↪ portability. By organizing resources into clusters, engineers could manage
↪ large-scale workloads efficiently, ensuring applications were resilient,
↪ scalable, and portable across various environments.

### The Impact (Meaning)
The introduction of Kubernetes clusters marked a transformative moment in
↪ cloud-native application deployment. Their strengths lie in enabling
↪ efficient management of containerized workloads at scale, supporting both
↪ on-premise and cloud deployments seamlessly. Clusters are essential for
↪ hosting applications that demand scalability and flexibility, allowing
↪ businesses to adapt quickly to changing demands without compromising
↪ performance.

The significance of Kubernetes clusters extends beyond just technical efficiency;
↪ they empower organizations to innovate and grow by providing a robust
↪ foundation for their digital infrastructure. While there might be
↪ complexities in setting up and managing these clusters initially, the
↪ benefits far outweigh any initial hurdles.

## 2. Storytelling Hooks

### Dramatic Question
"Can a group of interconnected nodes transform how businesses deploy applications
↪ across diverse environments?"

### Point of View
From the perspective of an engineer facing the challenge of scaling containerized
↪ applications efficiently.

## 3. Classroom Delivery Tips

### Pacing
- **Pause** after introducing the problem to let students reflect on the
↪ challenges faced before Kubernetes clusters.
- **Ask a question**: "What do you think are some potential issues with manually
↪ managing large-scale containerized applications?"
- **Pause** again after explaining the 'Aha!' moment, allowing students to
↪ consider how clusters change the game for application deployment.

### Analogy
```

```
Imagine a cluster as a team of superheroes (nodes), each with unique skills and
↪  abilities. Individually, they can perform well, but together, they form an
↪  unstoppable force capable of handling any challenge thrown their way-whether
↪  it's battling villains in public spaces, protecting private sanctuaries, or
↪  operating across both worlds seamlessly. Just like this superhero team,
↪  Kubernetes clusters work collectively to manage complex tasks efficiently and
↪  effectively, ensuring that applications run smoothly no matter where they're
↪  deployed.

### Interactive Activities for Clusters
### 1. Debate Topic

**Statement:** "Clusters are indispensable for Kubernetes in managing large-scale
↪  containerized workloads efficiently, given their ability to support both
↪  on-premise and cloud deployments; however, the absence of any identified
↪  weaknesses raises concerns about potential oversight or unexplored
↪  limitations."

This topic invites debate by highlighting the undeniable strengths of clusters
↪  while encouraging participants to critically assess whether a lack of
↪  identified weaknesses is a genuine indication of perfection or an area that
↪  requires deeper investigation.

### 2. What If Scenario Question

**Scenario:** Imagine you are the lead architect at a tech company planning to
↪  deploy a new microservices-based application. Your team can choose between
↪  using Kubernetes clusters for this deployment, which promises efficient
↪  management of large-scale containerized workloads and flexibility across both
↪  on-premise and cloud environments, or opting for an alternative orchestration
↪  tool that has been recently updated with enhanced security features but lacks
↪  the robust scalability support provided by Kubernetes clusters.

**Question:** If you were to choose between deploying your application using
↪  Kubernetes clusters or the alternative orchestration tool, how would you
↪  justify your decision? Consider both the strengths of using clusters and the
↪  implications of choosing an option with no identified weaknesses. How might
↪  potential unknown limitations affect your choice?

This scenario encourages students to apply their understanding of the concept by
↪  weighing the trade-offs between efficiency and scalability against security
↪  enhancements and the unknown risks associated with a system perceived as
↪  flawless.
```

```
---

## Teaching Module: Master Components
## 1. The Story (Problem -> Solution -> Impact)

### The Problem (Event)
In a bustling city of containerized applications called Clusteropolis, chaos
↪  reigned. Without any centralized control, containers were like unruly
↪  citizens-some wandered aimlessly without resources, while others crowded
↪  specific areas, causing system strain and inefficiencies. Applications would
↪  sporadically fail because there was no one to ensure they had what they
↪  needed or to manage their health. The city's infrastructure was in disarray,
↪  struggling under the weight of uncoordinated operations.

### The 'Aha!' Moment (Experience)
Amidst this chaos, a visionary engineer named Alex discovered a set of guidelines
↪  that could bring order to Clusteropolis: the Master Components. These were
↪  akin to the mayor and city council of the Kubernetes cluster-a team
↪  responsible for overseeing everything from resource allocation to health
↪  monitoring.

The Master node emerged as the central hub of control, equipped with key
↪  components:
- **API Server**: Acting like a communication center, it received requests and
↪  ensured that every decision was properly logged.
- **Scheduler**: This component was akin to an urban planner, determining where
↪  each container should reside for optimal performance.
- **Controller Manager**: Like a diligent city manager, it maintained the desired
↪  state of operations by monitoring and rectifying any discrepancies.

Together, these components worked in harmony to enforce laws and policies across
↪  Clusteropolis, ensuring resources were used efficiently and applications ran
↪  smoothly.

### The Impact (Meaning)
With Master Components at the helm, Clusteropolis transformed. Applications
↪  flourished with consistent performance, and resource utilization became
↪  optimized. This centralized control allowed for uniform decision-making
↪  processes that ensured stability and reliability across the entire city.
```

```
The strengths of these components lay in their ability to provide a cohesive
↪  management system, making them indispensable for orchestrating complex
↪  container operations efficiently. While there were no notable weaknesses
↪  presented by Alex's discovery, the trade-off was clear: relinquishing some
↪  autonomy at individual container levels for the greater good of cluster-wide
↪  harmony and efficiency.

## 2. Storytelling Hooks

- **Dramatic Question**: "How can a single central authority bring order to a
↪  chaotic city of containers?"

- **Point of View**: "From the perspective of Alex, an engineer tasked with
↪  revitalizing Clusteropolis, a city of containerized applications."

## 3. Classroom Delivery Tips

- **Pacing**:
  - Pause after describing the initial chaos in Clusteropolis to let students
  ↪  visualize the problem.
  - Ask, "What do you think happens when there's no central control?" before
  ↪  introducing Master Components.
  - After explaining each component (API Server, Scheduler, Controller Manager),
  ↪  pause for a moment of reflection on their specific roles.

- **Analogy**:
  - Compare Kubernetes Master Components to a city government. The API Server is
  ↪  like the main communication office receiving all requests and dispatching
  ↪  information. The Scheduler acts as the urban planner deciding where new
  ↪  buildings (containers) should go, while the Controller Manager ensures that
  ↪  everything runs according to plan, much like a city manager overseeing
  ↪  daily operations and ensuring compliance with policies.

This storytelling approach will help students understand the importance of Master
↪  Components in Kubernetes by relating them to real-world organizational
↪  structures.

### Interactive Activities for Master Components
### 1. Debate Topic

**Debate Statement:**
"Centralized control in Master Components is essential for effective cluster
↪  management despite having no identified weaknesses."
```

```
**Discussion Points:**

- **For**: Centralized decision-making ensures uniformity, reduces configuration
↪  errors, and simplifies administrative tasks.
- **Against**: The absence of recognized weaknesses might suggest a lack of
↪  critical scrutiny or potential oversights that have not yet been encountered.

### 2. 'What If' Scenario Question

**Scenario:**
Imagine you are the IT manager for a rapidly growing tech company. Your team is
↪  considering implementing Master Components to manage your expanding data
↪  cluster, which will handle sensitive customer information and
↪  mission-critical applications. You know that Master Components offer
↪  centralized control over management tasks. However, during discussions, some
↪  team members express concerns about potential unknown weaknesses.

**Question:**
If you were to implement Master Components in this scenario, how would you
↪  justify your decision given their strength of providing centralized control,
↪  yet acknowledging the lack of identified weaknesses? What precautions or
↪  additional strategies might you employ to mitigate any unforeseen issues that
↪  could arise from undiscovered vulnerabilities?

**Considerations for Discussion:**

- The importance of centralized management for consistency and reliability.
- Strategies to monitor and identify potential weaknesses as they emerge.
- Balancing reliance on Master Components with contingency planning.


---

## Teaching Module: Kubelets
## 1. The Story (Problem -> Solution -> Impact)

### The Problem (Event)
```

```
In the bustling city of Techville, applications needed to run seamlessly across
↪  numerous servers in a sprawling data center. However, managing these
↪  containers manually was becoming chaotic and inefficient. Servers were
↪  overloaded, some remained underutilized, and ensuring every application ran
↪  as expected required constant human intervention. This led to frequent
↪  downtimes, frustrating both users and administrators.

### The 'Aha!' Moment (Experience)
One day, a brilliant engineer named Alex stumbled upon an ingenious idea while
↪  observing ants in the park. Just like how each ant knows its role in
↪  maintaining the colony's efficiency, what if each server had an agent that
↪  could manage itself? This inspired Alex to create "Kubelets"-agents that
↪  would run on every node (server) within the Kubernetes cluster.

These Kubelets were designed to communicate with the Master components of the
↪  cluster. They received instructions and managed the lifecycle of containers
↪  on their respective nodes, ensuring each container ran smoothly as intended.
↪  By doing so, they maintained the desired state of applications without
↪  requiring constant oversight from human administrators.

### The Impact (Meaning)
The introduction of Kubelets revolutionized how Techville's data center operated.
↪  Applications began running more reliably and efficiently across all servers.
↪  With decentralized management, resources were allocated effectively, allowing
↪  for effortless scaling as demand fluctuated. This meant less downtime and
↪  happier users!

Kubelets empowered the cluster to self-manage, freeing engineers from mundane
↪  tasks and enabling them to focus on innovation. While there were no
↪  significant weaknesses identified, this approach marked a pivotal shift in
↪  container orchestration, highlighting the significance of efficient resource
↪  management at the node level.

## 2. Storytelling Hooks

- **Dramatic Question**: "How can we make our data center smarter without adding
↪  more human oversight?"

- **Point of View**: Narrate from Alex's perspective as an engineer who is both
↪  challenged by and inspired to solve Techville’s server management issues.

## 3. Classroom Delivery Tips
```

```
- **Pacing**:
  - Pause after introducing the problem in Techville to allow students to ponder
  ↪  the challenges faced without Kubelets.
  - Ask a question: "What do you think would happen if we could automate managing
  ↪  these servers?"
  - Slow down during the explanation of how Kubelets work, using simple language
  ↪  and reinforcing key points.
  - Reflect on the impact after describing it, encouraging students to consider
  ↪  its implications in real-world scenarios.

- **Analogy**:
  - Compare Kubelets to a diligent manager in an office. Just as a manager
  ↪  ensures each employee is productive, has their tasks completed, and
  ↪  resources are used efficiently without needing constant reminders from
  ↪  higher-ups, Kubelets ensure containers on their nodes run smoothly and
  ↪  report back to the central system for oversight.

This storytelling module helps convey the concept of Kubelets in a relatable and
↪  engaging manner, making it easier for students to grasp its significance and
↪  functionality.

### Interactive Activities for Kubelets
### Debate Topic:

**Statement:** "Kubelets are essential for efficient scaling and resource
↪  allocation in containerized environments due to their decentralized
↪  management capabilities; however, this strength is moot without any
↪  identified weaknesses."

**Debate Directions:**

- **Affirmative Position:** Argue that the strengths of Kubelets-specifically
↪  their ability to enable decentralized management, which leads to efficient
↪  scaling and resource allocation-are significant enough to overshadow any
↪  hypothetical or latent weaknesses. Emphasize how these strengths can
↪  transform container orchestration by enhancing performance, reducing
↪  bottlenecks, and improving system resilience.
```

```
- **Negative Position:** Challenge the idea that the absence of current
↪  identified weaknesses makes Kubelets' strengths unequivocally beneficial.
↪  Discuss potential hidden drawbacks such as complexity in configuration,
↪  potential for mismanagement at scale, or security concerns that might arise
↪  with decentralized systems. Argue that without acknowledging possible future
↪  weaknesses, reliance on their current strengths could lead to unforeseen
↪  challenges.

### What If Scenario Question:

**Scenario:** Imagine you are the CTO of a rapidly growing tech startup. Your
↪  company is scaling its services globally and has decided to adopt Kubernetes
↪  for container orchestration. You have two options: use Kubelets for
↪  decentralized management or rely on a centralized management system that
↪  offers similar functionalities but with slightly higher latency in resource
↪  allocation.

**Question:**

If you choose to implement Kubelets, what potential benefits can your
↪  organization expect from their decentralized management capabilities?
↪  Conversely, how would you address any unspoken concerns about
↪  decentralization, considering no explicit weaknesses are currently
↪  identified?

**Guidelines for Discussion:**

- **Benefits Analysis:** Discuss the advantages of using Kubelets in terms of
↪  scalability, resource efficiency, and fault tolerance. Consider how these
↪  strengths can support your company's growth and improve service delivery.

- **Unspoken Concerns:** Explore potential challenges that might arise from
↪  decentralization, such as complexity in managing multiple nodes, potential
↪  security vulnerabilities due to distributed control, or issues with
↪  consistency across different environments.

- **Justification:** Justify your choice by weighing these trade-offs. Consider
↪  factors like the company's current infrastructure, team expertise, and
↪  long-term strategic goals. Discuss how you would mitigate any risks
↪  associated with decentralization while maximizing the benefits of Kubelets.
```