Decoding the Epistemic Climate: An Analysis of Engineering Course Materials Using Natural Language Processing

Mitchell Gerhardt
Virginia Tech
Ph.D Student – Department of Engineering Education
M.S. Student – Department of Computer Science
mitchg@vt.edu

Abstract

Epistemic cognition – how individuals understand, evaluate, and justify knowledge and knowing processes – plays a critical role in engineering education. While prior research has examined how teachers and classroom interactions shape students' epistemic cognitions, less attention has been paid to how course materials like textbooks contribute to the epistemic climate of engineering education. This study introduces a novel approach to analyzing knowledge presentation in engineering textbooks using natural language processing (NLP). We develop a taxonomy of ten distinct knowledge types (e.g., conceptual, procedural, ethical) and train a transformer-based model to identify these types in computer science textbook passages. Using a synthetic dataset of 10,000 labeled examples, our model achieves strong classification performance with macro-averaged F1-scores of 0.79, demonstrating that different knowledge types have distinctive textual characteristics. Analysis reveals significant patterns in how knowledge types co-occur, with strong connections between conceptual and epistemic knowledge but less integration of mathematical and procedural knowledge. The results provide empirical evidence for theoretical frameworks of epistemic climate while offering a scalable method for analyzing how engineering knowledge is presented to students. This work has important implications for understanding and intentionally shaping the epistemic messages conveyed through engineering course materials.

1 Introduction

Individuals' experiences and backgrounds shape their beliefs and attitudes toward knowledge and knowing processes, with education serving as a central incubator for their acquisition and application (Hofer). Such beliefs and attitudes are often defined as an individual's epistemic cognitions, or how they understand, evaluate, and justify knowledge and knowing processes (Greene et al.). These cognitions encompass beliefs about the nature, source, and justification of knowledge, as well as the dispositions, goals, and costs individuals adopt toward knowing and knowing. While they have been described as stage-like developmental sequences, sets of independent beliefs, and metacognitive resources (Hofer and Pintrich), epistemic cognition plays a central role in how individuals acquire and apply information, making them critically important to engineering education (Faber and Benson).

Engineers constantly deal with complex, ill-structured problems. These problems require evaluating multiple sources of information, making value judgments under uncertainty, and decision-making within technical and non-technical constraints (Goldman). Accordingly, engineers must develop sophisticated epistemic cognitions that value flexibility, advanced information-seeking behaviors, and unique application-based knowledge (Kelly). Many of these cognitions are formed in educational environments as students engage in classrooms and with course materials (Greene et al.; Chinn et al., 2011). These knowledge artifacts and processes, like pedagogies, curriculum materials, and assessments, contribute to a classroom or discipline's "epistemic climate" – the "facets of knowledge and knowing that are salient in a learning or educational environment, that interact with and influence a learner's epistemic [cognitions]" (Muis et al., p. 335). Epistemic climates expose students to normative ways of describing, valuing, presenting, and applying knowledge in their disciplines, enculturing them into disciplinary ways of knowing and shaping their epistemic cognitions.

Muis et al. () describe five components of an educational epistemic climate, referred to as PACES: Pedagogy, Authority, Curriculum, Evaluation, and Support. Pedagogy refers to classroom teaching approaches, focusing on what teachers do and say, including the types of questions they ask and an-

swer. Authority addresses the role of the teacher as an authority figure, and Curriculum describes the institutional content, materials, and resources used in classrooms. Assessment practices, including feedback and the connections between answers and knowledge, are represented by Evaluation, and Support refers to the scaffolding teachers provide during learning and conceptual change processes. The authors argue that aligning these components can foster more rapid and lasting epistemic change than approaches focusing on isolated elements. However, current engineering education research has emphasized classroom interactions and pedagogical approaches, presenting an opportunity to investigate how course materials contribute to the epistemic climate in engineering classrooms.

Textbooks have long been a source of similar inquiries in education. In science education, studies have analyzed how science textbooks present the nature of scientific knowledge, often finding that they present science as bodies of facts rather than as processes of inquiry and experimentation (Abd-El-Khalick et al.). Similarly, history textbooks vary in handling conflicting historical accounts and present historical knowledge differently (e.g., as fixed narratives vs. interpretive processes) (Repoussi and Tutiaux-Guillon; Stearns et al., 2000). Mathematical textbooks also emphasize procedural knowledge over conceptual understanding in proofs, prioritize imitative reasoning, and mathematics as a fixed body of knowledge (Herbel-Eisenmann; Lithner; Stylianides). Common across these findings is a tendency toward absolutist and static presentations of knowledge, directly dichotomizing learning objectives to broaden students' conceptual understanding and potentially alienating students in the process (Danielak et al.; Montfort et al.).

Many of these studies have relied on close reading and coding to identify epistemic themes in textbooks and course materials (Bock; Weinbrenner). Due to the time-intensive nature of this analysis, however, they tend to be limited to a few textbooks or courses. Additionally, there are challenges in maintaining consistency in qualitative coding across textbooks as researchers may not have the domain experience to recognize unique disciplinary language, particularly for complex argumentative or explanatory structures in text. Limited samples and resource-intensive methods can also make results challenging to compare across contexts and periods – the coding scheme from one textbook may not be applied to another or across time. Quantitative methods have used frequency counts of certain terms or concepts to analyze content. However, these approaches may rely on specific word sequences or language, omitting semantically similar language and reducing complex ideas to simple, countable units. These word sequences may cover broad patterns, but miss deeper, more nuanced meanings and how ideas develop and connect throughout a text.

Considering the challenges associated with traditional thematic analysis of textbooks, we propose a methodological framework leveraging natural language processing (NLP) techniques to analyze knowledge presentation in engineering textbooks. This study focuses on developing and validating this methodology through three main contributions: (1) establishing a theoretically-grounded taxonomy of knowledge types in computer science education, (2) developing a synthetic dataset generation approach to create training data for knowledge type classification, and (3) demonstrating the feasibility of using transformer-based models to distinguish between different forms of knowledge presentation. By focusing on an element of the Curriculum component of the PACES model (Muis et al.), we aim to validate methods that could eventually enable systematic analysis of how knowledge is presented in engineering textbooks, thereby shaping students' epistemic cognitions and identities. The following research question guides this study: "How effectively can NLP-based approaches distinguish between different forms of knowledge presentation in computer science educational materials?"

2 Defining Knowledge Types

Our classification framework builds on established knowledge taxonomies in engineering education, identifying distinct knowledge categories common in engineering textbooks. A knowledge type refers to a distinct category of information or understanding that serves a particular purpose in engineering education and practice. These knowledge types reflect different aspects of what students need to know and understand. For example, conceptual knowledge represents theoretical understanding, like what stress and strain mean, while practical knowledge might apply conceptual concepts and show how a stress-strain relationship may be used to design a safe structure. Textbooks often use many such knowledge types throughout the material, weaving them together through examples, case

studies, and foundational lessons. Often, it is this multifaceted approach to conveying knowledge that defines engineering textbooks – an intricate mix of theory and applications mirroring engineering practice.

We defined knowledge types as having three distinct qualities:

1. Knowledge types have a distinct purpose – Each knowledge type serves specific educational and professional goals. It has characteristic learning outcomes and contributes differently to engineering understanding and functions. For example, conceptual knowledge builds the theoretical understanding necessary for engineering analysis and reasoning. This differs from practical and professional knowledge which develops the capabilities to solve real engineering problems and work effectively in the profession. These knowledge types are compatible and may be connected within a textbook lesson but are distinct in their goals and purposes. This premise recognizes institutional definitions of engineering knowledge. For instance, the National Society of Professional Engineers (NSPE) outlines an Engineering Body of Knowledge that includes capabilities such as Mathematics, Natural Sciences, Design, and Communication, each serving unique roles in professional practice (noa).

2. Knowledge types have characteristic ways they are expressed in text – Specific vocabulary, terminology, sentence structures, and rhetorical patterns indicate different knowledge types (joh). Conceptual knowledge might use abstract and theoretical language, mathematical expressions, universal statements ("In all cases..."), or particular definitional structures ("X is defined as..."). This contrasts practical and professional knowledge which might use action-oriented language, conditional statements ("If X occurs, then..."), real-world references, and procedural descriptions. These variations are critical because they enable NLP techniques to learn and differentiate between knowledge types.

3. Knowledge types have different implications for how students understand what counts as knowledge in engineering – Different knowledge types present different implications for students' understanding, including what counts as knowledge in engineering, how knowledge is justified, how certain/uncertain knowledge claims are, and who has authority over knowledge (Michalaka and Giogli). Conceptual knowledge may imply that knowledge is based on mathematical and scientific principles, inscribing a high degree of certainty and deriving authority from proofs. However, practical and professional knowledge may imply that knowledge is validated through successful application. Consequently, multiple solutions may be correct, requiring engineers to balance uncertainty and trade-offs. Through this process, authority comes from professional experience, standards, and application viability. Thus, knowledge types convey different messages about the source, certainty, connectedness, and justifications of engineering knowledge, shaping students' learning outcomes and understanding of the profession (Muis et al.).

Our structured approach to defining engineering knowledge types serves two important purposes. First, from a computational perspective, these well-defined categories provide clear boundaries and characteristics that NLP models can learn to recognize. This enables both the generation of high-quality synthetic training data and the development of classification models that can reliably distinguish between different forms of knowledge presentation in educational texts. Second, from an educational research perspective, this framework enables systematic analysis of how knowledge is conveyed in engineering education. By quantifying the relative prevalence and distribution of different knowledge types, we can examine potential biases in how engineering knowledge is presented to students, identify gaps in knowledge coverage, and understand the implicit messages being conveyed about what types of knowledge are valued in the field.

3 Determining Engineering Knowledge Types

This definition guided our development of ten distinct knowledge types using a combination of manual analysis and generative AI techniques. First, we analyzed available CS textbooks to find similarities in their content, including chapter/section structure patterns, content presentation, common transition

elements, and recurring educational components. The textbooks surveyed are discussed in the following sections but are also listed in Table 2. Approximately 30 passages drawn from these textbooks were then given to Claude, the large generative model from Anthropic along with the following prompt:

> You are an expert social science researcher studying computer science textbooks. Given a collection of engineering textbook passages, let's develop a framework for categorizing them into distinct knowledge types. For each passage, consider:
>
> 1. Content characteristics:
>   (a) What is the primary purpose of this passage?
>   (b) What information is being conveyed?
>   (c) How is it being presented?
> 2. Linguistic features:
>   (a) What vocabulary and phrasing is used?
>   (b) What sentence structures appear?
>   (c) What discourse patterns are present?
>
> After examining several passages, identify emerging patterns that suggest distinct knowledge categories. For each proposed category:
>
> 1. Define its distinguishing characteristics
> 2. Explain how it differs from other categories
> 3. Provide example passages that typify this category
>
> Test your categories by attempting to classify new passages. If you find many passages that could fit multiple categories, refine your framework to make the categories more distinct. Please analyze the provided passages and propose a categorization framework.

This method initially yielded 14 codes, which was reduced to ten according to the criteria established in Section 2. This reduced set is shown in Table 1. We used a zero-shot approach based on similar NLP-based qualitative coding techniques in engineering education research (Katz et al.), which yield codes similar to those performed by manual human coding. Although we did not perform human coding to verify the correctness of these themes, we analyzed the model output to verify knowledge type distinctness and coherence. Future research should continue exploring the efficacy of these tools, determining whether NLP-based techniques for qualitative analysis constitute valid alternatives while identifying potential limitations in their application.

4 Generation Pipeline

To develop and evaluate our in-progress classification system, we first needed to address the challenge of limited labeled training data. While real computer science textbooks are readily available, manually annotating passages with their knowledge types would be prohibitively time-consuming and potentially inconsistent across different annotators. To overcome these limitations, we developed a structured data generation approach using large language models (LLMs). By carefully crafting prompts that specified the desired computer science topic, textbook context, location, and target knowledge types, we were able to generate synthetic textbook passages with known labels. This approach allowed us to create a diverse, balanced dataset spanning different knowledge categories while maintaining control over the distribution and combination of knowledge types. The synthetic dataset serves dual purposes: providing training data for our fine-tuned classification model and establishing a benchmark for evaluating the model's performance across different knowledge categories and textbook contexts.

Accordingly, we generated diverse examples of textbook content across different computer science topics, contexts, and textbook locations. This data generation process employed careful controls to ensure an authentic representation of knowledge types. Each generated excerpt was contained by:

1. The specific computer science topic (e.g., Software Design, Data Structures)

2. The textbook context (e.g., main text, example, exercise)

3. The location within the text (e.g., section beginning, after equation)

Code	Abbrev	Definition
Conceptual Knowledge	CON	Core theoretical principles and fundamental concepts that form the basis of engineering understanding, including: basic theories and principles that underpin the engineering discipline, explanations of key engineering concepts and how they relate to each other, fundamental laws and equations central to the field, and abstract models and frameworks used to understand engineering phenomena
Historical Knowledge	HIS	Information about the development of engineering concepts, techniques, and technologies over time
Procedural Knowledge	PRO	Step-by-step explanations of problem-solving methods, experimental procedures, or design processes
Interdisciplinary Knowledge	INTER	Connections between the specific engineering discipline and other fields
Epistemic Knowledge	EPIS	Information about how knowledge is constructed, validated, and evolves within the engineering field
Metacognitive Knowledge	META	Guidance on how to approach learning and problem-solving in engineering
Ethical Knowledge	ETH	Discussion of ethical considerations in engineering practice and research
Mathematical Knowledge	MATH	Equations, derivations, and mathematical models used in engineering
Uncertainty and Limitations	UNC	Discussions about the limitations of current knowledge and areas of ongoing research or debate in the field
Practical and Professional Knowledge	PRAC	The application of engineering concepts in real-world contexts, including the use of current technologies and adherence to professional standards and practices. This encompasses case studies and real-world problem-solving scenarios, information about current engineering technologies, tools, and software, professional standards, codes, and best practices in engineering, and practical design processes and decision-making in engineering projects

Table 1: Types of Engineering Knowledge and Their Definitions

4. One or more target knowledge types from our taxonomy

This methodology enabled the creation of a rich, labeled dataset capturing the natural complexity of computer science educational texts while maintaining precise knowledge type annotations. The resulting corpus serves as both training data for our fine-tuned model and a resource for studying knowledge representation in computer science materials.

However, the use of synthetic data in this study naturally raises questions about its representativeness of actual engineering textbooks. Our approach rests on two key assumptions. First, we posit that LLMs, having been trained on extensive collections of text that include academic materials, can effectively replicate the language patterns, terminology, and structural conventions found in engineering textbooks. Second, we believe that with carefully designed prompts and appropriate contextual constraints, the generated passages can capture the essential features needed for knowledge type classification. To validate these assumptions, future work could pursue several directions. A comparative analysis between synthetic and real textbook passages could help quantify the degree of similarity in terms of linguistic features, knowledge representation, and structural characteristics. Additionally, investigating whether LLMs can perform this classification task through prompt engineering alone would provide insights into the feasibility of zero-shot approaches. Such research could build on recent work in NLP-based qualitative thematic analysis (Katz et al.), potentially offering new methodologies for evaluating and validating synthetic training data in educational text analysis.

4.1 Identifying Topics

Our definition of knowledge types implies that these characteristics should be consistent across different computer science (CS) educational materials. This external consistency aligns with research on educational epistemic climates by Muis et al. (2016) and Feucht (2010), who draw from Ecological Systems Theory (EST) (Bronfenbrenner, 2009) to show how disciplines develop their own epistemic norms and practices. While some variation exists within CS education, analyzing multiple CS courses and textbooks enables us to identify common patterns in how knowledge is presented to students.

To determine different knowledge types, we began by examining CS textbooks from the Virginia Tech undergraduate curriculum. We identified the primary textbook for each required course in the curriculum, as shown in Table 2. When digital versions were unavailable, we used comparable alternative editions that were accessible online. We extracted passages from these textbooks to use as inputs for the process described in Section 3. While these initial samples served only to identify knowledge types, our future work will apply the resulting trained model to analyze the complete content of these textbooks. In summary, we used CS textbooks to identify knowledge types, used these types to create labeled training data for an NLP-based model, and plan to use the trained model to analyze the original textbook content.

4.2 Prompt Design

Our prompt development process involved multiple iterations to address challenges with the generated content. Throughout this process, we consistently used the qwen2.5:14b-instruct model (Team, 2024), which was selected because it was large enough to follow complex prompt instructions (unlike smaller models) while remaining computationally efficient for large-scale generation. While this approach produced realistic outputs with sufficient variation, larger models might generate even more authentic content – a hypothesis that future research could test by comparing outputs across models of different sizes.

The initial prompt focused on generating multiple examples for a single knowledge type, requiring strict JSON formatting and including basic guidelines for authenticity and variety. However, these initial experiments showed that generating authentic textbook passages required providing more contextual information to the model. Without this context, the generated samples tended to use repetitive patterns and formulaic introductions (e.g., "In this passage..." or "[Topic X] is an approach that..."). While such patterns can appear in textbooks, they don't reflect the full diversity of textbook writing styles. To address this limitation, we enhanced our prompts by incorporating specific contexts and passage locations. We used the Claude model to help generate potential contexts, and after refinement, we identified the key contexts and locations shown in Tables 3 and 4.

However, yet this approach still proved inadequate – there was too much regularity between out-

Course Name	Textbook
CS 1114 – Intro to Software Design	OpenDSA CS 1114 Online Textbook – Intro to Software Design
CS 2114 – Software Design & Data Structures	OpenDSA CS 2114 Online Textbook – Software Design & Data Structures
CS 2104 – Intro to Problem-Solving for CS	Whimbey, A., Lochhead, J., & Narode, R. (2013). *Problem solving and comprehension* (7. ed). Routledge.
CS 2505 – Intro to Computer Organization I	Patt, Y. N., & Patel, S. J. (2004). *Introduction to computing systems: From bits and gates to C and beyond* (2. ed). McGraw-Hill Higher Education.
CS 2506 – Intro to Computer Organization II	Patterson, D. A., & Hennessy, J. L. (2021). *Computer organization and design: The hardware/software interface* (Sixth edition). Morgan Kaufmann.
CS 3114 – Data Structures and Algorithms	OpenDSA CS 3114 Online Textbook – Data Structures & Algorithms
CS 3214 – Computer Systems	Bryant, R. E., & O'Hallaron, D. R. (2016). *Computer systems: A programmer's perspective* (Third edition). Pearson.
CS 3604 – Professionalism in Computing	Spier, R. (Ed.). (2002). *Science and technology ethics*. Routledge.
CS 3304 – Comparative Languages	Sebesta, R. W. (2019). *Concepts of programming languages* (Twelfth edition). Pearson.

Table 2: Computer Science courses and textbooks

puts, particularly those sharing a knowledge type and context/location. To remedy this, we modified the prompt to handle multiple knowledge types simultaneously while simplifying the output structure to generate single examples rather than multiple ones. While this improved the quality of generated content, we still observed repetitive patterns and themes in the outputs. Subsequent prompt versions incorporated more specific guidance about content placement and context within topics, explicitly discouraged common introductory patterns, and introduced the concept of assumed prior knowledge – allowing passages to begin as if continuing from previous content. We also added clarifications about both content placement and writing style to enhance contextual authenticity.

The final prompt version (shown in the Appendix) provides more detailed guidance for natural writing style and location-appropriate content. This version achieved a better balance between sophisticated textbook writing and proper implementation of knowledge types. Overall, our iterative prompt development process demonstrated a progression toward generating more authentic content that better reflects actual engineering textbook presentations.

4.3 Generation

Using the finalized prompt, we generated a sample of 10,000 labeled textbook passages. This operation was conducted on an Apple M2 Ultra Mac Studio equipped with a 24-core CPU, a 60-core GPU, a 32-core Neural Engine, 192 GB of unified memory, and 8 TB of SSD storage. Despite the advanced hardware, the process took approximately three days. While generating such a dataset manually would require significantly more time, this duration is still substantial, particularly in light of concerns about the environmental impact of generative AI technology (van Wynsberghe, 2021; Wu et al., 2022). We take these concerns seriously and implemented several measures during the generation process to ensure output validity and correctness, minimizing the need for additional rounds of generation. Table 7 in the Appendix shows a few examples generated by the aforementioned process.

4.4 Summary

To address the challenge of limited labeled training data for classifying knowledge types in computer science textbooks, we developed a data generation pipeline using LLMs. Through iterative prompt refinement, we generated a dataset of 10,000 labeled passages, ensuring diversity across computer

Content Type	Description
Theoretical Discussion	In-depth exploration of concepts
Mathematical Derivation	Step-by-step mathematical development
Proof	Logical development of a mathematical or theoretical proof
Scenario Analysis	Detailed examination of specific situations
Worked Example	Step-by-step problem solution
Practical Application	Real-world implementation details
Case Study	Specific real-world examples and analysis
Algorithm Description	Detailed explanation of algorithmic steps
Experimental Procedure	Description of laboratory or testing methods
Design Process	Steps in engineering design methodology
Problem Solving	Systematic approach to solving a specific problem
Comparison Analysis	Contrasting different approaches or technologies
Historical Development	Discussion of how concepts evolved over time
Implementation Details	Specific technical details of implementation
Literature Review	Discussion of current research and findings
System Architecture	Description of component relationships
Optimization Process	Steps to improve or optimize a solution
Failure Analysis	Examination of system failures or limitations
Integration Discussion	How components or concepts work together
Performance Analysis	Evaluation of system or method performance
Trade Off Analysis	Examining competing factors or decisions
Requirements Analysis	Discussion of system or design requirements
Simulation Description	Details of modeling and simulation approaches
Data Analysis	Statistical or analytical examination of data
Debugging Process	Systematic approach to finding and fixing issues
Validation Process	Methods for verifying results or designs
Cross Disciplinary Application	Applications in other fields
Future Directions	Discussion of emerging trends or research areas

Table 3: Types of context types

Section Type	Description
Section Beginning	Opening of a new section
Section Middle	Middle of a section's main content
Section End	Concluding part of a section
Subsection Beginning	Start of a subsection
Subsection Middle	Middle of a subsection
Subsection End	End of a subsection
Paragraph Beginning	Start of a new paragraph
Paragraph Middle	Middle of an ongoing paragraph
Paragraph End	Conclusion of a paragraph
After Equation	Text following a mathematical expression
After Figure	Text following a diagram or figure
After Example	Text following a worked example
Before Exercise	Lead-in to practice problems
Sidebar	Supplementary information box

Table 4: Types of passage locations

science topics, contexts, and knowledge types. The final dataset serves as both training data for a classification model and a benchmark for evaluating its performance. While the generation process leveraged advanced hardware and sophisticated prompts, it raised concerns about the environmental impact and representativeness of synthetic data, which we tried to mitigate through careful validation and optimization.

5 Initial Model Training

We utilized the synthetic dataset generated in the previous steps to train a model capable of identifying and classifying distinct knowledge types in CS textbooks. To achieve this, we began by developing a robust preprocessing pipeline to clean, tokenize, and structure the textbook data, ensuring consistency across diverse sources. We then employed a transformer-based model, which was fine-tuned using the annotated examples reflective of different knowledge types. This model was trained to recognize nuanced patterns in the presentation of knowledge, guided by carefully designed labels and embeddings. By iteratively refining the dataset and model, we aim to build a tool that not only accurately classifies knowledge types but also provides insights into the pedagogical strategies embedded within engineering textbooks. The overarching goal is to deepen our understanding of the epistemic climate in engineering education, offering a foundation for improving teaching materials and fostering a richer learning experience.

5.1 Loading the Data

To initiate the analysis, we began by systematically loading textbook data into the program. The data consisted of the generated passages which were uploaded to Github for easier distribution and stored in a structured format compatible with our preprocessing pipeline. We utilized Python for its versatility in handling large textual datasets and integrated libraries like `pandas` to organize the data into a tabular structure, allowing efficient manipulation and analysis. Each passage was tagged with metadata, such as the chapter and section it originated from, to preserve context and facilitate downstream tasks.

To ensure consistency, we applied a uniform encoding scheme, converting all text into UTF-8 format and stripping extraneous characters or formatting artifacts. This step was critical for maintaining compatibility with subsequent tokenization and embedding processes. Additionally, we implemented checks to handle missing or incomplete data, ensuring that only high-quality passages were included in the training and testing datasets. The result was a clean, well-structured corpus of textbook excerpts, ready for the next stages of our pipeline.

5.2 Preprocessing the Data

The preprocessing pipeline was designed to transform the raw textbook passages into a format suitable for training and evaluation within a ML framework. The first step involved tokenizing the text data using a transformer-based tokenizer, specifically tailored to align with the pre-trained language model employed in this study, BERT (Devlin et al., 2019). Tokenization split the passages into subword units while preserving the semantic structure and ensuring compatibility with the model's input requirements.

Following tokenization, we generated input features, including `input_ids`, `token_type_ids`, and `attention_mask`, which encode the tokenized sequences, distinguish between segments in multi-segment inputs, and mask padding tokens during attention computations, respectively. To retain the structural and epistemic context of the passages, we also included metadata embeddings for associated labels, such as chapter, section, and topic. Next, we applied text truncation or padding to standardize input length to the model's maximum sequence capacity (e.g., 512 tokens). This ensured uniformity across batches during training. Additionally, we employed label encoding to convert the annotated epistemic categories into multi-label vectors, reflecting the potentially overlapping nature of knowledge types in the passages.

Then, to prepare for training, the dataset was split into training, validation, and test sets using a stratified sampling strategy. This maintained the proportional representation of knowledge types across subsets, ensuring robustness during evaluation. Data augmentation techniques, such as synonym replacement and paraphrasing, were also explored to enrich the dataset and enhance model generalization.

5.3 Model Setup

The model architecture was based on the BERT pre-trained transformer, chosen for its proven capability in capturing contextualized representations of text (Devlin et al., 2019). Fine-tuning was performed to adapt the model to the task of knowledge type classification. The architecture consisted of the base transformer model, followed by a custom classification head. This head included a dense layer, dropout for regularization, and a final output layer with a sigmoid activation function to handle multi-label classification.

For optimization, we used the AdamW optimizer with a learning rate schedule incorporating linear warmup and decay, tailored to the dynamics of transformer fine-tuning (You et al., 2019). The loss function was binary cross-entropy, chosen to accommodate the multi-label nature of the task (Rezaei-Dastjerdehei et al., 2020). Training was conducted in mini-batches, leveraging GPU acceleration to handle the computational demands of processing large sequences. We also employed techniques such as gradient clipping to stabilize training and mitigate exploding gradients (Zhang et al., 2019), as well as early stopping to prevent overfitting (Ying, 2019). Evaluation metrics included precision, recall, and F1-score for each knowledge type, with macro-averaged scores providing an overall assessment of model performance. By integrating domain-specific preprocessing and fine-tuned model adaptations, our setup ensured that the model was both accurate and capable of capturing the nuanced epistemic characteristics of engineering education texts.

5.4 Training Steps

The training process was designed to fine-tune a transformer-based model for the task of multi-label classification, identifying various knowledge types

within textbook passages. The following steps outline the process with Table 5 showing the hyperparameters used.

Hyperparameter	Value
Optimizer	AdamW
Learning Rate	5e-5 (initial)
Learning Rate Scheduler	Linear warmup and decay
Batch Size	16
Dropout Rate	0.1
Loss Function	Binary Cross-Entropy
Epochs	Up to 10 (with early stopping)
Gradient Clipping Threshold	1.0
Validation Frequency	End of each epoch
Maximum Sequence Length	512 tokens
Early Stopping Patience	3 epochs (no improvement)
Evaluation Metrics	Precision, Recall, F1-Score (Macro)

Table 5: Hyperparameters used in the fine-tuning process

5.4.1 Dataset Preparation

After preprocessing, the dataset was split into training, validation, and test subsets, ensuring proportional representation of the annotated knowledge types across all sets. The training set was used for model optimization, the validation set for hyperparameter tuning and early stopping, and the test set for final evaluation. All subsets were batch-processed to ensure efficient use of computational resources.

5.4.2 Model Initialization

We initialized the BERT pre-trained transformer model with a custom classification head. The classification head was designed to handle the multi-label classification task and consisted of a fully connected dense layer with sigmoid activation. The pre-trained weights served as a starting point, capturing general language features, which were fine-tuned for our domain-specific task.

5.4.3 Training Configuration

1. **Optimizer**: We employed the AdamW optimizer, known for its compatibility with transformer architectures, with weight decay to mitigate overfitting.

2. **Learning Rate Scheduling**: A learning rate scheduler with linear warmup and decay was implemented to ensure smooth convergence, especially in the initial epochs where learning stability is critical (Kim et al., 2021).

3. **Loss Function**: Binary cross-entropy was used as the loss function, enabling effective handling of the multi-label nature of the task.

5.4.4 Training and Regularization

The training loop was executed over multiple epochs, where each epoch comprised a sequence of batch-level forward and backward passes. During the forward pass, tokenized sequences were input into the transformer model to generate logits corresponding to each knowledge type. These logits were then compared with the ground truth labels using a binary cross-entropy loss function, allowing the computation of batch-level loss. A backward pass followed, where gradients were propagated through the network, and the model's parameters were updated using the AdamW optimizer. To ensure stability and prevent issues such as exploding gradients, gradient clipping was applied during this step. Regularization was incorporated through dropout layers in the classification head, which introduced stochasticity into the training process and helped mitigate overfitting.

5.4.5 Monitoring and Evaluation

To monitor training progress, validation checkpoints were implemented at the end of each epoch. The model's predictions on the validation set were compared against the annotated labels, and metrics such as precision, recall, and F1-score were calculated for each knowledge type. These metrics guided the fine-tuning of hyperparameters and informed early stopping decisions, where training was halted if validation performance stagnated for a predefined number of epochs. Once training was complete, the model that achieved the best validation performance was evaluated on the test set. This final evaluation involved calculating per-label precision, recall, and F1-scores, as well as a macro-averaged F1-score to provide an overall measure of the model's performance. Additionally, confusion matrices were generated to analyze classification errors, offering insights into overlapping or misclassified knowledge types.

5.4.6 Post-training Analysis

: Finally, the trained model was analyzed for interpretability. Attention weights from the transformer were visualized to identify key words or phrases influencing predictions. This analysis provided

insights into how the model recognized and categorized knowledge types, aligning with our broader epistemic cognition goals.

6 Results

The model's performance, as evidenced by the evaluation metrics, demonstrates a strong ability to classify the different knowledge types in engineering education textbooks. Across all samples, the model achieved a micro F1-score of 0.7871, a macro F1-score of 0.7937, and a weighted F1-score of 0.7886, indicating balanced performance across both common and less frequent labels. Precision and recall values were similarly strong, with a micro precision of 0.7719 and a micro recall of 0.8029.

The detailed per-class evaluation shown in Table 6 revealed the model's nuanced handling of different knowledge types. Categories such as ETH (Ethics) and HIS (Historical) exhibited exceptionally high F1-scores of 0.968 and 0.86, respectively, possibly due to their distinct semantic patterns and sufficient representation in the dataset. On the other hand, categories like MATH (Mathematics) and PRO (Procedural knowledge) displayed relatively lower F1-scores (0.684 and 0.683, respectively), likely due to semantic overlaps with other categories or more complex textual representations. The model's thresholds for each class were optimized to balance precision and recall. Notably, PRAC (Practical Knowledge) had the lowest threshold (0.55), suggesting that it required a more inclusive approach to detect its instances, while ETH and HIS maintained higher thresholds of 0.8, reflecting their distinct and identifiable textual characteristics.

As shown in Figure 1, the label co-occurrence matrix highlighted significant overlaps between certain labels, such as CON (Conceptual knowledge) and EPIS (Epistemic reflection) or META (Metacognition) and PRAC, which points to the inherent multi-label nature of the task. This overlap was consistent with the epistemic complexity of textbook passages, where sentences often embody multiple dimensions of knowledge. Confidence analysis provided additional insights into the model's predictions. For example, CON (Conceptual knowledge) had the highest mean confidence (0.480) and a relatively narrow confidence distribution, indicating consistent predictions. Conversely, labels like ETH and HIS exhibited lower mean confidence scores (0.272 and 0.269, respectively) but higher precision, suggesting that the model made more conservative yet accurate predictions for these classes. The bar chart on Per-Class Metrics emphasizes the precision-recall balance across all knowledge types, while the Optimal Thresholds by Class highlights the distinct requirements for each category. The Label Co-Occurrence Matrix vividly showcases the interdependencies between categories, and the Mean Prediction Confidence by Class underscores the varying degrees of certainty in the model's outputs.

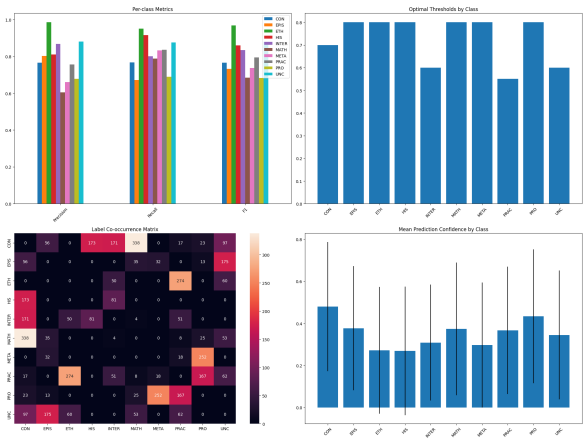

Figure 1: Initial model training results

7 Discussion

In this section, we discuss the results of both the generation and prediction pipelines, encompassing connections to epistemic climate analysis, knowledge type relationships, implications for engineering education, and methodological and theoretical contributions. Ultimately, the NLP-based techniques utilized in this research represents a preliminary step toward future investigations assessing multiple dimensions of the epistemic climate in computer science.

7.1 Epistemic Climate Analysis

The model's ability to effectively distinguish between different knowledge types in computer science textbooks provides empirical support for Muis et al.'s () theoretical framework of epistemic climate in engineering education. The high F1-scores achieved across most knowledge type categories (particularly ETH at 0.968 and HIS at 0.86) suggest that these categories represent meaningfully distinct ways of presenting knowledge in educational texts. This quantitative approach to analyzing the curriculum component of the PACES framework

Label	Threshold	Precision	Recall	F1-Score	Support
CON (Conceptual)	0.7	0.766	0.767	0.766	665
EPIS (Epistemic)	0.8	0.802	0.672	0.732	296
ETH (Ethics)	0.8	0.985	0.951	0.968	284
HIS (Historical)	0.8	0.811	0.916	0.860	215
INTER (Interdisciplinary)	0.6	0.869	0.801	0.834	297
MATH (Mathematical)	0.8	0.604	0.789	0.684	279
META (Metacognitive)	0.8	0.661	0.833	0.737	215
PRAC (Practical)	0.55	0.756	0.836	0.794	434
PRO (Procedural)	0.8	0.678	0.689	0.683	421
UNC (Uncertain)	0.6	0.881	0.876	0.879	380

Table 6: Per-class performance metrics, including thresholds, precision, recall, F1-scores, and support (number of instances) for each knowledge type.

offers new insights into how knowledge is structured and presented to students. The distribution patterns of knowledge types may reveal implicit messages about what kinds of knowledge are valued in computer science education. For instance, the high precision in identifying ethical and historical knowledge suggests these types have distinctive characteristics, perhaps indicating their treatment as specialized or supplementary content rather than being integrated throughout the curriculum.

7.2 Knowledge Type Relationships

The co-occurrence patterns revealed in our analysis suggest that certain knowledge types naturally cluster together in computer science education materials. The strong relationship between conceptual (CON) and epistemic (EPIS) knowledge types indicates that textbooks often couple theoretical understanding with reflections on the nature of that knowledge. This coupling may serve to scaffold students' development of both content knowledge and epistemological understanding. However, the lower performance in classifying mathematical (MATH) and procedural (PRO) knowledge types (F1-scores of 0.684 and 0.683 respectively) suggests these categories are more contextually embedded and harder to isolate. This finding aligns with research on the integrated nature of mathematical and procedural knowledge in engineering education (Engelbrecht et al., 2012), where these forms of knowledge often appear intertwined with conceptual explanations and practical applications.

7.3 Educational Implications

Our analysis provides several important implications for engineering education practice. First, the ability to quantitatively assess the distribution of knowledge types in educational materials offers a new tool for curriculum development and evaluation. Authors and instructors can use these insights to create more balanced learning materials that intentionally incorporate diverse knowledge types. Second, the identification of co-occurrence patterns between knowledge types can inform more effective sequencing of educational content. For example, the strong coupling between conceptual and epistemic knowledge suggests that explicit discussion of knowledge construction might best accompany the introduction of new theoretical concepts. Finally, the varying confidence levels in knowledge type classification (from 0.480 for conceptual knowledge to 0.272 for ethical knowledge) may indicate areas where the presentation of certain knowledge types could be made more explicit or distinctive.

7.4 Methodological Contributions

This study makes several methodological contributions to engineering education research. First, it demonstrates the feasibility of using NLP techniques to analyze epistemic aspects of educational texts at scale. The successful application of transformer-based models to this task opens new possibilities for large-scale analysis of educational materials. Second, our synthetic data generation approach provides a novel solution to the challenge of developing training data for epistemic analysis. This method could be adapted for other educational research contexts where labeled data is scarce. Finally, our multi-label classification approach acknowledges and accounts for the complexity of how knowledge is presented in authentic educa-

tional materials, providing a more nuanced view than single-label approaches.

7.5 Theoretical Implications

The success of our model in distinguishing between knowledge types provides empirical support for the theoretical framework outlined in Section 2. The distinct textual characteristics we identified for each knowledge type suggest that these categories represent meaningful differences in how knowledge is presented in engineering education. However, the significant overlap between certain knowledge types, as shown in our co-occurrence analysis, suggests that epistemic cognition may be more integrated than previously theorized. This finding aligns with recent work suggesting that students' epistemic development involves learning to navigate multiple, interconnected forms of knowledge rather than progressing through discrete stages or categories.

8 Limitations

The current prediction model exhibits several methodological limitations that warrant careful consideration when attempting to generalize its classification capabilities across diverse computer science educational materials. A fundamental constraint emerges from the training dataset's composition, which demonstrates notable class imbalances and relies on a relatively modest sample size of approximately 9,000 training examples. This limited corpus may inadequately capture the full spectrum of knowledge representations present in computer science pedagogy. The dataset's construction methodology also presents epistemological challenges, as it lacks established inter-rater reliability measures and comprehensive thematic analysis protocols that would ensure exhaustive coverage of potential knowledge types. The absence of such validation mechanisms introduces potential systematic biases in knowledge classification. Furthermore, the model's architectural framework, while leveraging BERT's natural language understanding capabilities, encounters constraints in processing extended textbook passages and may oversimplify the inherently interconnected nature of computer science knowledge types. The predefined taxonomic structure, though theoretically grounded, potentially fails to accommodate emergent or hybrid forms of knowledge representation. Additional limitations stem from potential sampling biases in the source materials, which may not fully represent the diverse pedagogical approaches and subject matter depth across computer science subfields. These methodological constraints suggest several promising directions for future research, including: expanding the training corpus through systematic sampling strategies, implementing robust inter-rater reliability protocols, developing more nuanced knowledge type representations, and incorporating domain adaptation techniques to enhance generalization capabilities. Such methodological refinements would strengthen the model's ability to serve as a reliable tool for automated knowledge type classification in computer science education materials.

9 Conclusion

This study introduces a novel approach to analyzing the epistemic climate in engineering education through automated analysis of course materials. By developing a taxonomy of knowledge types and leveraging NLP techniques, we demonstrate that different forms of knowledge presentation in computer science textbooks can be reliably identified and analyzed at scale. Our results show high classification performance across multiple knowledge types, with F1-scores ranging from 0.968 for ethical knowledge to 0.683 for procedural knowledge, validating both our theoretical framework and methodological approach. This work makes several key contributions: (1) it provides empirical evidence for distinct knowledge types in engineering education materials, (2) it demonstrates a scalable method for analyzing epistemic climate through curriculum materials, and (3) it reveals patterns in how different types of knowledge are presented and interrelated in computer science education. These findings have important implications for textbook authors, instructors, and researchers working to understand and improve how engineering knowledge is conveyed to students. Future work should explore the application of these methods to other engineering disciplines, investigate temporal changes in knowledge presentation across different editions of textbooks, and examine how different patterns of knowledge presentation relate to student learning outcomes. Through continued development of such analytical approaches, we can better understand and intentionally shape the epistemic climates we create in engineering education.

References

Cambridge handbook of engineering education research.

Professional engineering body of knowledge.

Fouad Abd-El-Khalick, John Y. Myers, Ryan Summers, Jeanne Brunner, Noemi Waight, Nader Wahbeh, Ava A. Zeineddin, and Jeremy Belarmino. A longitudinal analysis of the extent and manner of representations of nature of science in u.s. high school biology and physics textbooks. 54(1):82–120. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/tea.21339.

Annekatrin Bock. Theories and methods of textbook studies. In Eckhardt Fuchs and Annekatrin Bock, editors, *The Palgrave Handbook of Textbook Studies*, pages 57–70. Palgrave Macmillan US.

Urie Bronfenbrenner. 2009. *Ecology of Human Development: Experiments by Nature and Design*. Harvard University Press, Cambridge.

Clark A. Chinn, Luke A. Buckland, and Ala Samarapungavan. 2011. Expanding the Dimensions of Epistemic Cognition: Arguments From Philosophy and Psychology. *Educational Psychologist*, 46(3):141–167.

Brian A. Danielak, Ayush Gupta, and Andrew Elby. Marginalized identities of sense-makers: Reframing engineering student retention. 103(1):8–44. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/jee.20035.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Johann Engelbrecht, Christer Bergsten, and Owe Kågesten. 2012. Conceptual and procedural approaches to mathematics in the engineering curriculum: Student conceptions and performance. *Journal of Engineering Education*, 101(1):138–162.

Courtney Faber and Lisa C. Benson. Engineering students' epistemic cognition in the context of problem solving. 106(4):677–709. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/jee.20183.

Florian C. Feucht. 2010. Epistemic climate in elementary classrooms. In Lisa D. Bendixen and Florian C. Feucht, editors, *Personal Epistemology in the Classroom*, 1 edition, pages 55–93. Cambridge University Press.

Steven L. Goldman. Why we need a philosophy of engineering: a work in progress. 29(2):163–176.

Jeffrey A. Greene, William A. Sandoval, and Ivar Bråten, editors. *Handbook of epistemic cognition*, first published edition. Educational psychology handbook series. Routledge.

Beth A. Herbel-Eisenmann. From intended curriculum to written curriculum: Examining the voice of a mathematics textbook. 38(4):344–369. Publisher: National Council of Teachers of Mathematics.

Barbara K. Hofer. Personal epistemology research: Implications for learning and teaching. 13(4):353–383.

Barbara K. Hofer and Paul R. Pintrich. The development of epistemological theories: Beliefs about knowledge and knowing and their relation to learning. 67(1):88–140. Publisher: American Educational Research Association QID: Q56047718.

Andrew Katz, Mitch Gerhardt, and Michelle Soledad. Using generative text models to create qualitative codebooks for student evaluations of teaching. 23:16094069241293283. Publisher: SAGE Publications Inc.

Gregory J. Kelly. Methodological considerations for the study of epistemic cognition in practice. In *Handbook of Epistemic Cognition*. Routledge. Num Pages: 16.

Chiheon Kim, Saehoon Kim, Jongmin Kim, Donghoon Lee, and Sungwoong Kim. 2021. Automated learning rate scheduler for large-batch training. *arXiv preprint arXiv:2107.05855*.

Johan Lithner. Mathematical reasoning in calculus textbook exercises. 23(4):405–427.

Dimitra Michalaka and Emanuele Giogli. Which skills are more important in the engineering world? perceptions of college students, recent graduates, and employers.

Devlin Montfort, Shane Brown, and Victoria Whritenour. Secondary students' conceptual understanding of engineering as a field. 3(2).

Krista R. Muis, Gregory Trevors, and Marianne Chevrier. Epistemic climate for epistemic change. In *Handbook of Epistemic Cognition*. Routledge. Num Pages: 29.

Maria Repoussi and Nicole Tutiaux-Guillon. New trends in history textbook research: Issues and methodologies toward a school historiography. 2.

Mohammad Reza Rezaei-Dastjerdehei, Amirmohammad Mijani, and Emad Fatemizadeh. 2020. Addressing imbalance in multi-label classification using weighted cross entropy loss function. In *2020 27th national and 5th international iranian conference on biomedical engineering (ICBME)*, pages 333–338. IEEE.

Peter N. Stearns, Peter Seixas, and Sam Wineburg. 2000. Knowing, teaching, and learning history : national and international perspectives.

Gabriel J. Stylianides. Reasoning-and-proving in school mathematics textbooks. 11(4):258–288. Publisher: Routledge _eprint: https://doi.org/10.1080/10986060903253954.

Qwen Team. 2024. Qwen2.5: A Party of Foundation Models! Section: blog.

Aimee van Wynsberghe. 2021. Sustainable AI: AI for sustainability and the sustainability of AI. *AI and Ethics*, 1(3):213–218.

Peter Weinbrenner. Methodologies of textbook analysis used to date. In *History and Social Studies*. Routledge. Num Pages: 14.

Carole-Jean Wu, Ramya Raghavendra, Udit Gupta, Bilge Acun, Newsha Ardalani, Kiwan Maeng, Gloria Chang, Fiona Aga, Jinshi Huang, Charles Bai, Michael Gschwind, Anurag Gupta, Myle Ott, Anastasia Melnikov, Salvatore Candido, David Brooks, Geeta Chauhan, Benjamin Lee, Hsien-Hsin Lee, Bugra Akyildiz, Maximilian Balandat, Joe Spisak, Ravi Jain, Mike Rabbat, and Kim Hazelwood. 2022. Sustainable ai: Environmental implications, challenges and opportunities. In *Proceedings of Machine Learning and Systems*, volume 4, pages 795–813.

Xue Ying. 2019. An overview of overfitting and its solutions. In *Journal of physics: Conference series*, volume 1168, page 022022. IOP Publishing.

Yang You, Jing Li, Sashank Reddi, Jonathan Hseu, Sanjiv Kumar, Srinadh Bhojanapalli, Xiaodan Song, James Demmel, Kurt Keutzer, and Cho-Jui Hsieh. 2019. Large batch optimization for deep learning: Training bert in 76 minutes. *arXiv preprint arXiv:1904.00962*.

Jingzhao Zhang, Tianxing He, Suvrit Sra, and Ali Jadbabaie. 2019. Why gradient clipping accelerates training: A theoretical justification for adaptivity. *arXiv preprint arXiv:1905.11881*.

A Appendix

A.1 Prompt for Sample Generation

```
You are an expert engineering educator
    with extensive experience in
    curriculum design across various
    engineering disciplines. Your task
    is to generate realistic
    textbook-style excerpts that
    exemplify one or more specific
    types of knowledge within a given
    engineering subject and topic. Your
    output will be in JSON format for
    easy parsing and processing.

Use the following inputs to guide your
    content generation:

Subject: {subject}
Topic: {topic}
Location: {location}
Content Type: {context}
Knowledge Types:
{knowledge_types}

Create a textbook-like excerpt of
    approximately 100-150 words that
    demonstrates one or more of the
    specified knowledge types within
    the context of the given subject
    and topic. Each excerpt should be
    written in a style appropriate for
    the specified context.

Guidelines for creating the excerpts:

1. Write naturally as if continuing an
    ongoing discussion - avoid
    repeatedly starting sentences with
    phrases like "In [field]..." or
    "When doing [task]..."

2. Match the writing style and content
    structure to the context: {context}
    -- {context_description}

3. Use varied sentence structures and
    openings. For example:
   - Direct statements about concepts
   - Descriptions of processes
   - Analysis of relationships
   - Explanations of phenomena
   - Discussion of implications

4. Match the writing style to the
    location: {location} --
    {location_description}. For example:
   - section_beginning: Frame new ideas
       without "In this section..."
   - paragraph_middle: Continue as if
       mid-discussion
   - after_equation: Reference the
       previous mathematical content
   [etc.]

5. Consider the natural flow of ideas
    based on the location:
   - If it's paragraph_middle, assume
       previous context
   - If it's after_figure, refer to
       visible elements
   - If it's subsection_end, synthesize
       rather than introduce

6. Maintain authenticity by using
    appropriate technical language and
    concepts relevant to the subject
    and topic.
7. Include relevant details such as
    theories, equations, or real-world
    applications as appropriate for the
    context.
8. Ensure that each excerpt clearly
    represents at least one of the
    specified knowledge types, but may
    include multiple types if
    appropriate.

Then, format your response as a JSON
    object with the following structure:

{{
```

```
  "subject": "{subject}",
  "topic": "{topic}",
  "example": "[Insert 100-150 word
      excerpt here]"
}}

Ensure that your output is valid JSON.
    Escape any special characters
    within the text fields as necessary.
```

A.2 Example Passages from Generation Pipeline

ID	Subject	Topic	Example	Codes	Context	Location
4764	Computer Science	Intro to Software Design	The system architecture of software design encompasses multiple layers and components, each with specific roles and interfaces. For instance, in a typical layered architecture, such as the Model-View-Controller (MVC), the model handles data management, the view is responsible for visual representation, and the controller mediates between user input and application processing. This separation of concerns facilitates modular design and improves maintainability. Historical developments have shown that advancements like MVC and microservices architecture emerged to address complexities in software systems over time, reflecting an evolution towards more scalable and flexible designs.	INTER, CON, HIS	System Architecture	Sidebar
7443	Computer Science	Intro to Computer Organization II	Recent research has emphasized the importance of mathematical models in understanding and optimizing computer system performance. For instance, queuing theory is often applied to analyze the behavior of instruction pipelines and memory hierarchies. Equations such as Little's Law ($L = \lambda W$) provide a fundamental relationship between the average number of jobs in the system (L), the arrival rate (λ), and the average time spent in the system (W). These mathematical tools not only help in predicting system performance but also guide design decisions to minimize latencies and maximize throughput.	MATH	Literature Review	Before Exercise
6362	Computer Science	Comparative Languages	Throughout the evolution of programming languages, each generation has attempted to address limitations and inefficiencies identified in its predecessors. For instance, while imperative languages like C offered fine-grained control over system resources, they lacked abstractions for managing complex data structures efficiently. This gap was partly filled by object-oriented languages such as Java, which introduced encapsulation, inheritance, and polymorphism. However, despite these advancements, debates continue about the most effective paradigms for developing scalable software systems. Research in language design now often focuses on functional programming and its potential to simplify concurrent processing, although challenges remain in achieving efficient runtime performance.	UNC	Historical Development	Paragraph Middle
5513	Computer Science	Intro to Computer Organization I	The von Neumann architecture, illustrated in Figure 1, exemplifies a foundational concept in computer organization where the central processing unit (CPU), memory, and input/output systems are interconnected. This design underpins modern computing but also finds applications in other engineering fields such as control systems and telecommunications. For instance, the principles of data flow and storage can be seen in digital signal processing, where signals are transformed into a format that can be efficiently processed by CPUs or specialized hardware like DSPs (Digital Signal Processors). This cross-disciplinary application underscores the importance of understanding core theoretical principles in computer organization for broader engineering contexts.	CON	Cross-Disciplinary Application	After Figure

Table 7: Example passages from the dataset