systems

MDPI

Article

Infrastructure Gaps in Social Media-Based Programming Education: A Large-Scale Analysis of Learner Support Needs and the Case for Technical Presence

Zhuoyuan Tang [1,2], Wei Wei [1], Kai Liang [1] and Chi Kin Lam [1,*]

[1] Faculty of Applied Sciences, Macao Polytechnic University, Macao, China; p2413134@mpu.edu.mo (Z.T.); weiwei@mpu.edu.mo (W.W.); p2424530@mpu.edu.mo (K.L.)
[2] School of Information Technology, Zhejiang Financial College, Hangzhou 310018, China
* Correspondence: cklamsta@mpu.edu.mo

Abstract

Social media platforms increasingly function as informal education systems for programming learning, yet the systemic support structures these environments provide remain poorly understood. We analyzed 40,004 comments from programming tutorial videos on a major social media platform (2016–April 2025) to identify patterns of learner support needs at scale. Using BERTopic, we identified twelve discussion themes. We then consolidated these themes into a learner-needs typology based on their dominant support functions: instructional-oriented needs, operational support needs, and knowledge-constructionneeds. We mapped this typology onto the Community of Inquiry (CoI) framework to assess its explanatory coverage. This mapping revealed a critical systemic gap. Operational support needs, covering environment configuration, tool integration, dependency management, and technical troubleshooting, constituted the largest category (44.53% of theme-level discourse), exceeding both knowledge-construction needs (28.42%) and instructional-oriented needs (26.95%). Learners repeatedly described these infrastructure-level challenges as disrupting their attempts to engage with content, execute code for testing ideas, and coordinate with peers, yet these operational readiness needs are not fully specified by CoI's traditional presences. Social presence did not emerge as a standalone theme at the topic-modeling level; rather, social cues were often embedded within task-oriented troubleshooting. Based on these findings, we propose Technical Presence as a context-sensitive extension to the CoI framework, defined as the extent to which a learning community enables operational readiness through accessible infrastructure support and collaborative troubleshooting. As an infrastructural support condition, Technical Presence supports operational readiness within tool-dependent, practice-based learning: when learners report infrastructure failure, the conditions for enacting instructional design, cognitive inquiry, and peer collaboration are correspondingly weakened. These findings carry implications for content creators, platform developers, and education system designers seeking to strengthen the infrastructural foundations of technology-enhanced learning at scale.

Keywords: Community of Inquiry (CoI); technical presence; social media-based learning; programming education; education systems; learner needs; topic modeling

Check for updates

Academic Editors: Ben Clegg, Reinhard Bernsteiner, Michael Kohlegger and Christian Ploder

Received: 14 April 2026
Revised: 4 June 2026
Accepted: 11 June 2026
Published: 15 June 2026

Copyright: © 2026 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license.

1. Introduction

Social media platforms have become vital venues for self-directed programming learning, functioning in practice as large-scale, informal education systems that serve millions of

learners outside traditional institutional structures. Yet the systemic support these environments provide, or fail to provide, remains underexplored. The Community of Inquiry (CoI) framework, a cornerstone of online education research, proposes that meaningful learning emerges from the interplay of three presences: teaching presence (instructional design, facilitation, and direct instruction), cognitive presence (meaning-making through sustained inquiry and reflection), and social presence (affective expression and interpersonal interaction) [1]. While the CoI survey has demonstrated validity across diverse contexts [2], programming education introduces unique technical challenges, such as environment setup, dependency management, and tool configuration, which represent operational hurdles distinct from purely cognitive or social needs. At the system level, it remains unclear whether the traditional tripartite CoI model adequately captures the infrastructure-level requirements that determine whether learners can proceed with executable practice in these environments. Programming education offers a particularly revealing case for examining this question. Learners must not only acquire conceptual knowledge but also configure and operate computing environments before they can enact instructional guidance through executable practice [3,4]. When these environments fail, learners often describe immediate disruptions in ways that are directly observable in comment data.

Social media platforms host vast repositories of such programming tutorials, and the comment sections beneath them serve as dynamic spaces for feedback and peer support [5]. Unlike structured LMS environments, these open forums capture spontaneous, in-situ struggles and the support mechanisms that emerge organically within informal education systems. These comments are especially informative in programming education because they capture both conceptual questions and technical configuration problems as they arise during practice. Research suggests that many tutorial systems fail to address these fundamental barriers [6], leaving learners to seek help in the comments.

Despite CoI's prominence, its application to the intersection of social media and programming education remains limited, leaving an open question as to whether its existing categories can account for the granular, tool-centric barriers learners encounter in these informal education systems.

To address this gap, we pose three research questions:

RQ1. What patterns of learner support needs emerge from comments on social media-based programming tutorials?

RQ2. How can these needs be organized into a learner-needs typology that captures their dominant support functions?

RQ3. To what extent does the existing CoI framework account for these needs, and which support functions, if any, remain under-specified?

This study contributes to theory, method, and practice. Theoretically, it identifies an explanatory gap in the CoI framework for tool-dependent learning environments and proposes Technical Presence as a context-sensitive extension to the CoI framework. We conceptualize Technical Presence as an infrastructural support condition that enables operational readiness and supports the enactment of teaching, cognitive, and social presences through executable practice. Methodologically, the study combines large-scale computational text analysis with comment-level manual validation to strengthen the empirical grounding of the thematic patterns and presence categorizations. Practically, it provides evidence-based guidance for instructors, content creators, and platform designers seeking to improve the infrastructural foundations of learner support on social media.

2. Literature Review

2.1. Comment-Centered Learning Research

Comments in social media environments can reflect learners' authentic needs and problem-solving processes. Compared with traditional methods such as surveys and interviews, comments reveal hidden concerns and authentic opinions expressed in natural settings [7,8]. To analyze these large datasets, researchers have increasingly turned to computational text analysis. Recent studies have used techniques such as opinion mining, text classification, and topic modeling to derive actionable themes for instructional improvement [9].

Topic modeling has become a core technique for identifying thematic patterns in learner discourse. Existing methods range from Latent Dirichlet Allocation (LDA) [10] to transformer-based approaches such as BERTopic [11]. Work in MOOC contexts shows that these approaches can identify urgent learner issues and reveal the factors behind them [12]. When applied to large comment corpora, topic modeling also helps characterize learners' interests and engagement patterns across social media and course-based discussions [13,14]. Among these approaches, BERTopic provides a transformer-based topic-modeling pipeline that embeds documents with pre-trained sentence models, clusters semantically similar texts, and extracts topic representations using class-based TF-IDF [11]. This approach is well suited to short, informal social-media comments: by grouping texts according to semantic similarity before extracting topic representations, it can capture similar learner concerns expressed in varied wording [12,13]. Given its effectiveness in revealing learner needs from unstructured discourse, we apply BERTopic to comments from programming tutorial videos and map the resulting themes to the CoI framework to examine how well it accounts for learner needs in social media-based programming education.

2.2. Applying the CoI Framework in Social Media-Based Education

The CoI framework, initially proposed by Garrison et al. [1], is used to interpret learners' support needs in online environments through three dimensions: teaching presence, social presence, and cognitive presence. As research has evolved, the framework has been expanded to capture a broader range of learner requirements. "Learning presence," introduced by Shea et al. [15,16], emphasizes learners' self-regulation needs in autonomous learning contexts, while "emotional presence," examined by Cleveland-Innes and Campbell [17] and Jiang and Koo [18], focuses on affective needs related to satisfaction and engagement. These developments indicate that CoI serves as a diagnostic framework for identifying the types of support learners require in online and blended learning settings [19].

Empirical studies further demonstrate the robustness of the framework. Cognitive presence is positively associated with learning outcomes [20], and CoI-informed course design has been associated with improvements in teaching quality [21]. Even in large-scale environments such as MOOCs, the three presences remain identifiable, though often in more fine-grained forms such as subfactors related to course organization, group affect, or problem resolution [22]. Platform characteristics also influence how support needs manifest in discourse. In structured forums, deeply threaded discussions make teaching, social, and cognitive support needs easier to observe and classify [23], whereas comments on video platforms tend to be brief and task-oriented. The framework has also been applied in systems engineering education; for example, Li et al. (2023) [24] found that organizational affiliations shaped social and cognitive presence in an online learning community, yet the role of technical infrastructure in enabling or constraining these presences was not examined. In summary, CoI is widely used to capture learner support needs across diverse online learning environments.

Together, these extensions and empirical applications have broadened the CoI framework beyond its original tripartite structure. Şen-Akbulut et al. [25] further proposed technology-related sub-dimensions distributed across the existing three presences, treating technology as a means of supporting teaching, cognitive, or social processes. However, these lines of work do not directly conceptualize the operational readiness of the technical environment as a distinct, community-level support condition. In social media-based programming education, learners operate in technology-mediated environments [26,27] where tool configuration, environment setup, and troubleshooting shape learners' ability to proceed with executable practice [3,4]. Whether these operational needs fit within the traditional CoI presences or represent a systemic support dimension that requires separate treatment remains an open question. This unresolved issue forms the theoretical motivation for the present study.

2.3. Barriers in Programming Education

Programming education introduces well-documented barriers that shape the support learners require, making it a useful context for examining how well the CoI framework accounts for such needs. These barriers span conceptual, pedagogical, and technical-operational dimensions.

Conceptual barriers involve developing algorithmic thinking, debugging strategies, and problem-solving skills beyond syntax [28]. Task complexity can overwhelm novices when it exceeds working-memory capacity, and tool use may add extraneous load [4]. Threshold concepts such as recursion and object-oriented abstraction present particular challenges [29,30]. While these difficulties align with cognitive presence, which emphasizes meaning-making and inquiry, CoI does not address the additional cognitive load introduced by tool interactions.

Pedagogical barriers concern how learners engage with programming tools and content. Modern IDEs, though powerful, are often difficult for beginners and treated as peripheral rather than as resources requiring explicit scaffolding [31]. To reduce this burden, researchers have developed novice-oriented or browser-based environments that standardize execution contexts [32,33]. These issues relate to teaching presence, yet they also show that instructional support cannot be delivered effectively until technical prerequisites are satisfied.

Technical-operational barriers shape whether learners can proceed with executable practice. Environment setup, dependency management, and tool operation represent persistent obstacles, and even with AI assistance learners may encounter non-executable code requiring diagnosis and adjustment [34,35]. Such barriers concern operational readiness rather than instructional or interpersonal processes and are not easily classified under CoI's three presences. This suggests that technical configuration and troubleshooting constitute a distinct, system-level dimension of programming learning that is not fully captured by traditional CoI categories.

3. Methods

3.1. Research Context

This study focuses on Bilibili, a major Chinese social media and video-sharing platform whose average daily active users reached 109.4 million in the second quarter of 2025 [36]. Beyond entertainment, Bilibili has become an important site for self-directed learning; prior research reports that over 200 million users engage in learning activities on the platform [37]. The platform hosts a large collection of Python programming tutorials, and their comment sections serve as active spaces where learners report challenges, seek help, and exchange solutions. Although the dataset analyzed in this study is China-based,

the difficulties reflected in learner discourse, such as conceptual misunderstandings and technical configuration issues, are similar to those documented in global research on novice programming learning.

3.2. Research Design and Analytical Framework

Informed by a systems perspective that views social media platforms as informal education ecosystems comprising learners, content creators, and technological infrastructure, our study followed the analytical workflow shown in Figure 1. After data collection and preprocessing, we first profiled the learning community using descriptive statistics. We then applied BERTopic to identify core discussion themes (RQ1). The twelve resulting themes were consolidated into a learner-needs typology based on their dominant support functions (RQ2). We evaluated the explanatory coverage of the CoI framework by mapping this typology onto the framework (RQ3). Finally, we conducted a comment-level validation using stratified sampling and independent coding of dominant and secondary presences to assess the reliability and boundary validity of the theme-level categorization (RQ2–RQ3).

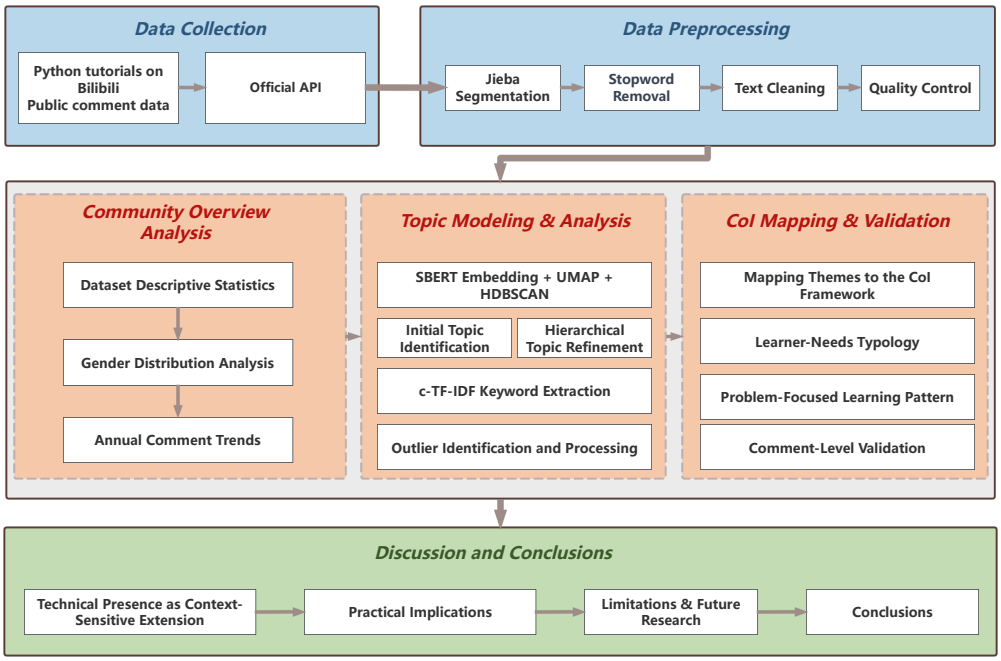


Figure 1. Analytical Workflow of the Study.

3.3. Data Collection and Sample Characteristics

Using Bilibili's official API, we collected publicly accessible comments from 290 Python tutorial videos posted between 2016 and April 2025. After removing advertisements, spam, and off-topic content through automated filtering, the analytical dataset comprises 40,004 comments from 32,211 unique commenters. The community composition is summarized as follows. Of these commenters, 66.19 percent did not disclose gender; 26.27 percent identified as male and 7.53 percent as female (about 3.5:1), a distribution consistent with prior reports on programming education demographics [38]. Annual comment volume increased steadily, with notable rises in 2020 (approximately 2.8 times the previous year) and in 2023–2024. All data were anonymized, derived from public sources, and processed in accordance with ethical research standards. Text preprocessing included Chinese word segmentation using Jieba and stopword removal. The stopword list combined several standard Chinese dictionaries (e.g., Harbin Institute of Technology) with domain-specific programming terms. All topic modeling and keyword extraction were conducted on the original Chinese comments; English translations are provided only for presentation.

3.4. Analytical Methods

3.4.1. BERTopic-Based Theme Identification and Structural Analysis

BERTopic [11] is a neural topic modeling pipeline that proceeds in four stages. First, each comment is converted into a dense vector using a pre-trained Sentence-BERT model [39], capturing semantic similarity beyond surface vocabulary. Second, the high-dimensional embeddings are projected into a lower-dimensional space via UMAP to make clustering computationally feasible while preserving local structure. Third, HDBSCAN identifies clusters of semantically similar comments without requiring a pre-specified number of clusters. Fourth, a class-based TF-IDF (c-TF-IDF) procedure extracts representative keywords for each cluster, producing interpretable topic labels. Compared with bag-of-words approaches such as LDA [10], this pipeline operates on contextual embeddings, making it better able to group comments that express similar concerns in varied wording.

We applied BERTopic to identify discussion topics in the collected comments. Recent studies in educational contexts have reported its usefulness for analyzing learner-generated content from social media and MOOCs [12,13]. To guide topic reduction and model selection, we evaluated topic configurations within the range of 15 to 40 topics. As shown in Figure 2, we selected a reduced topic configuration of K = 30 because it provided the best trade-off between two coherence metrics: C_V continued rising with larger K, while C_NPMI peaked at K = 25 and declined thereafter. Thus, K = 30 maintained both within-topic coherence and topic distinctiveness.

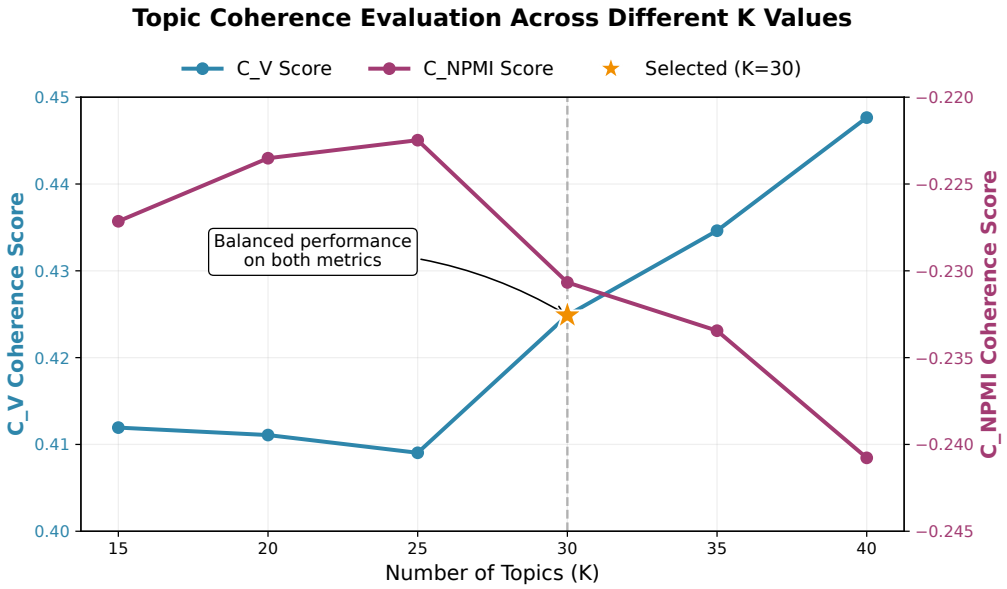


Figure 2. Topic Coherence Evaluation Curve.

We used the Sentence-BERT model paraphrase-multilingual-MiniLM-L12-v2 for embedding, applied UMAP for dimensionality reduction, and performed HDBSCAN clustering with min_cluster_size set to 65. This configuration produced 89 fine-grained topics. We then merged these into 30 secondary topics using the reduce_topics procedure. The reduce_outliers procedure flagged 46 semantically isolated comments; we retained them in corpus-level counts but did not assign them to higher-level themes. Figure 3 shows hierarchical clustering with three major semantic branches, which provided empirical support for subsequent consolidation. All analyses were implemented in Python 3.8.0 using BERTopic 0.16.2 (which integrates the UMAP and HDBSCAN steps above) and Jieba 0.42.1 for Chinese word segmentation.

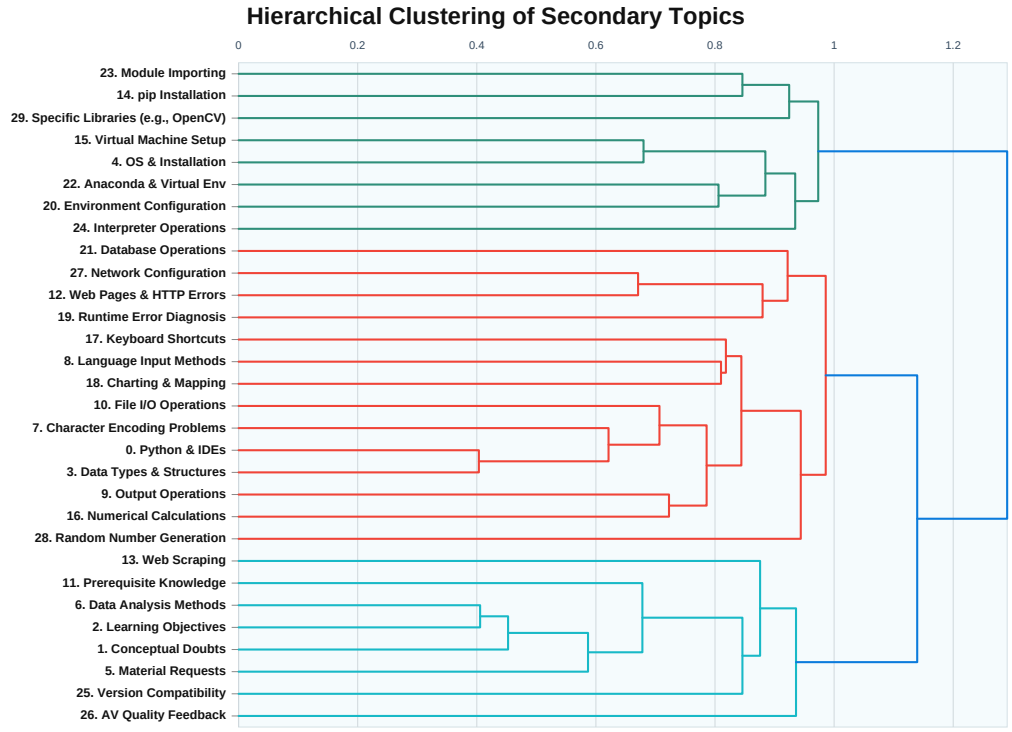


Figure 3. Hierarchical Clustering of Themes.

Finally, two researchers conducted semantic validation by inspecting the intertopic distance map and representative keywords. Through iterative discussion, the 30 secondary topics were consolidated into 12 stable primary topics. We extracted keywords with c-TF-IDF and examined their weight distributions to assess topic quality. As shown in Figure 4, most topics had core terms concentrated among the top-ranked words, suggesting clear semantic focus across distinct discussion areas.

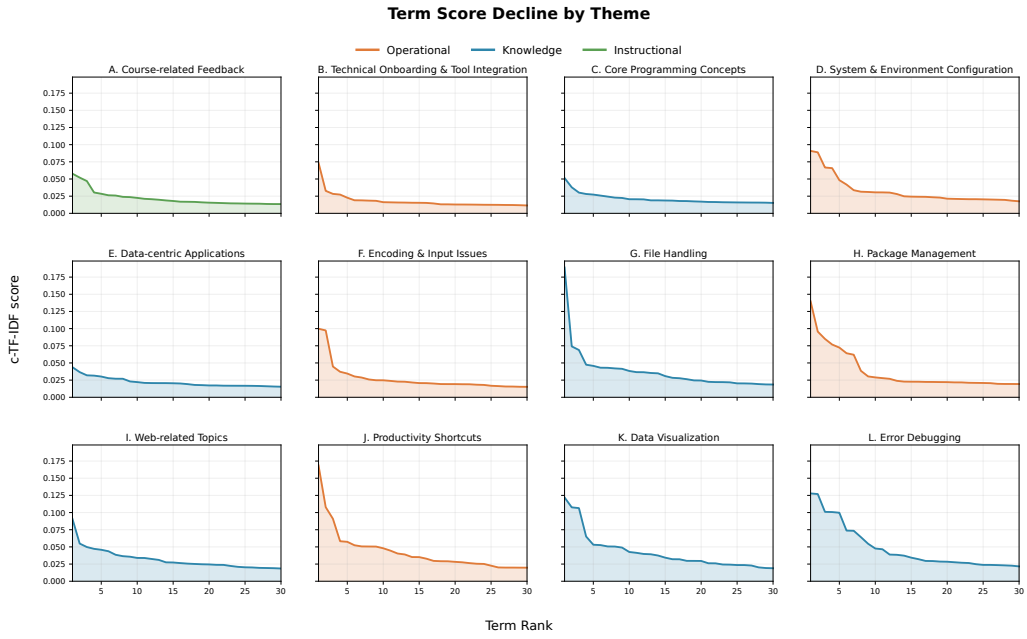


Figure 4. Decline in term scores by theme.

To move from topic-level findings to a characterization of learner needs, we introduced an intermediate consolidation step after theme identification. For each of the twelve primary themes, the two researchers examined its constituent sub-topics, representative keywords,

and representative comments to determine the theme's dominant support function in the learning process. Through iterative discussion, the twelve themes were consolidated into a small number of learner-needs categories.

3.4.2. Mapping Topics to the CoI Framework

Two researchers conducted qualitative coding to map the 12 identified themes onto the CoI framework [1,40]. During initial coding, several themes matched these three presences. However, some themes focused mainly on technical infrastructure and operational support needs, such as environment setup, tool configuration, and dependency management, which did not fall cleanly under teaching, cognitive, or social presence. Using inductive qualitative content analysis [41,42], we initially labeled these themes as "Others" to reflect patterns outside the traditional CoI framework. To more accurately describe their operational and infrastructural focus, we subsequently termed this category "Technical-Operational." In line with CoI coding practices that identify the primary communicative function of an utterance [43], we applied a dominant-intent rule: comments whose primary intent concerned technical setup, tool operation, or system configuration were coded as Technical-Operational. Table 1 provides representative examples based on dominant intent.

Table 1. Coding examples for CoI presence identification.

CoI Presence	Comment Example	Coding Rationale
Teaching Presence	"The pace is too fast; could you provide practice files?"	Instructional design, organization, and materials
Cognitive Presence	"What's the difference in use between lists and tuples?"	Conceptual inquiry and meaning-making
Social Presence	"As a beginner here, I truly feel welcomed."	Affective expression and identity projection
Technical-Operational	"VSCode will not recognize my virtual environment, and the interpreter path keeps resetting."	Infrastructure setup and tool-configuration requirements needed to proceed with learning activities

The two researchers achieved initial agreement on 10 of 12 themes (83.33%). The two disagreements were resolved through discussion and joint review of representative comments from each theme until reaching full consensus.

This mapping procedure enabled us to assess not only individual-level support needs but also the systemic adequacy of existing theoretical frameworks for capturing the full range of support functions observed in the education ecosystem.

3.4.3. Comment-Level Validation

While the theme-level mapping established correspondence between discussion themes and CoI categories based on aggregate semantic characteristics, individual comments within the same theme may exhibit heterogeneous presence patterns. To assess the boundary validity of the theme-level mapping at the individual-comment level, we conducted a stratified validation using Garrison et al.'s [1] foundational definitions and established CoI coding practices [44].

From the 12 primary themes, we drew a balanced stratified random sample of 300 comments (25 per theme). Because the sample was balanced across themes rather than proportionally drawn from the full corpus, it was used to assess coding reliability and boundary validity rather than to estimate corpus-level prevalence. Two researchers independently assigned each comment a dominant presence label (Teaching, Cognitive, Social,

or Technical-Operational) using the coding scheme in Table 2. The dominant-intent rule required coders to identify the primary communicative function of each comment; when multiple presence indicators co-occurred, the coder selected the one that best characterized the comment's central purpose. Technical-Operational was used as the empirical coding label for comments whose dominant function concerned operational readiness; Technical Presence refers to the theoretical construct proposed from this empirical pattern.

The coding scheme operationalized each presence type as follows. Teaching Presence encompasses course design, facilitation, and direct instruction [44]. Cognitive Presence involves learners' construction and confirmation of meaning through inquiry [45]. Social Presence refers to participants' ability to project themselves socially and emotionally in a community of inquiry [40,43]; we distinguished dominant Social Presence, which requires purposeful relationship-building as the primary intent, from brief social cues (e.g., "thanks") that were coded only as secondary presences. Technical-Operational was structured into three sub-dimensions: Infrastructure Configuration (environment setup, software installation, version management), Tool & Dependency Operation (package management, IDE configuration, dependency resolution), and Technical Troubleshooting (error diagnosis, compatibility issues, access and permission problems). For boundary cases, tool-selection advice in which the environment was assumed to be functioning was coded as Teaching, whereas operational failure reports were coded as Technical-Operational.

Table 2. Coding Scheme for Comment-Level Presence Identification.

Presence Type	Operational Indicators	Example Comment	Coding Notes
Teaching Presence	Course design/organization; instructional clarity/pacing; material/resource requests; facilitation feedback	"The explanation is too fast; could you provide the practice code?"	Code as Teaching when the primary intent addresses instructional design, delivery, or facilitation
Cognitive Presence	Conceptual inquiry; "why/how" questions; problem-solving; comparison of approaches; knowledge integration	"What's the difference between lists and tuples in terms of memory usage?"	Code as Cognitive when the primary intent involves meaning-making, conceptual understanding, or knowledge construction
Social Presence	Self-introduction; community building; affective expression as primary intent; identity projection; interpersonal connection	"Hi everyone! New learner here, excited to join this community and learn together."	Code as dominant Social Presence only when relationship-building is the primary intent. Brief social cues in task-focused comments are secondary only
Technical-Operational	*Infrastructure Configuration:* environment setup, software installation, version management. *Tool & Dependency Operation:* package management, IDE configuration, dependency resolution. *Technical Troubleshooting:* error diagnosis, compatibility issues, access/permission problems	"pip install fails with externally-managed-environment error on Ubuntu 24."	Code as Technical-Operational when the primary intent concerns system-level barriers to code execution or environment functionality

To capture within-comment complexity, coders also recorded secondary presence labels when clearly identifiable additional communicative functions appeared beyond the dominant code. This dominant/secondary coding procedure allowed us to identify co-occurring presence indicators and distinguish primary communicative functions from secondary social cues.

Inter-coder agreement was assessed using Cohen's kappa for dominant presence labels, with interpretation guided by Landis and Koch [46]. Reliability was calculated for dominant labels only; secondary presence labels were used descriptively to examine overlap among presences. Disagreements were resolved through discussion and joint review until consensus was reached.

4. Results

4.1. Thematic Patterns in Learner Comments

Topic modeling of 40,004 comments identified 12 primary themes (Table 3). Discussion was highly concentrated: the top four themes, including Course-related Feedback (26.95%), Technical Onboarding & Tool Integration (26.40%), Core Programming Concepts (14.33%), and System & Environment Configuration (8.39%), accounted for 76.07% of all comments. The remaining themes formed a long tail covering data applications, file handling, and productivity skills. These distributions show a learner focus on instructional quality, technical entry barriers, conceptual clarification, and the infrastructure challenges that participants described as disrupting their progress.

Table 3. Hierarchical Taxonomy of Learning Themes (*N*= 40,004; includes 46 outlier comments in cluster −1).

First-Level Theme	Count n (%)	Second-Level Topics (Original ID, Count n)	Top Keywords
A. Course-related Feedback	10,783 (26.95%)	1. Conceptual Doubts (4524); 2. Learning Objectives (3914); 5. Material Requests (2165); 26. AV Quality Feedback (180)	Focus: Explanation clarity, pacing, resource completeness, AV accessibility; Keywords: teacher, course, video, question, materials, audio, subtitle, explanation
B. Technical Onboarding & Tool Integration	10,560 (26.40%)	0. Python & IDEs (9133); 11. Prerequisite Knowledge (879); 24. Interpreter Operations (284); 25. Version Compatibility (264)	Focus: IDE selection, interpreter setup, version management, entry-level tool configuration; Keywords: Python, PyCharm, function, learning, py-file, download, interpreter, IDLE, version
C. Core Programming Concepts	5731 (14.33%)	3. Data Types & Structures (3785); 9. Output Operations (1243); 16. Numerical Calculations (581); 28. Random Number Generation (122)	Focus: Core data types/structures and basic I/O/numerical operations; Keywords: string, list, dictionary, set, variable, object, print, number, random

Table 3. *Cont.*

First-Level Theme	Count n (%)	Second-Level Topics (Original ID, Count n)	Top Keywords
D. System and Environment Configuration	3355 (8.39%)	4. OS & Installation (2508); 15. Virtual Machine Setup (359); 20. Environment Configuration (306); 27. Network Configuration (182)	Focus: OS/VM installation, environment variables, network/SSH connectivity; Keywords: download, install, version, Mac, Linux, Windows, VM, Ubuntu, PATH, IP, server
E. Data-centric Applications	3103 (7.76%)	6. Data Analysis Methods (2247); 13. Web Scraping (564); 21. Database Operations (292)	Focus: Data analysis, web scraping, database operations; Keywords: data, algorithm, programming, crawler, pandas, MySQL, database, computer
F. Encoding & Input Issues	2467 (6.17%)	7. Character Encoding Problems (1491); 8. Language Input Methods (976)	Focus: Text encoding and console/input handling issues; Keywords: code, VS Code, run, UTF, encoding, error, Chinese, input, garbled
G. File Handling	1057 (2.64%)	10. File I/O Operations (1057)	Focus: File paths, read/write operations, save/open; Keywords: file, document, folder, write, create, open, read, save
H. Package Management	935 (2.34%)	14. pip Installation (395); 22. Anaconda & Virtual Env (251); 23. Module Importing (170); 29. Specific Libraries (e.g., OpenCV) (119)	Focus: Dependency installation, environment creation, import errors; Keywords: pip, jieba, install, anaconda, jupyter, import, module, opencv
I. Web-related Topics	753 (1.88%)	12. Web Pages & HTTP Errors (753)	Focus: Web page access and HTTP error handling; Keywords: HTML, webpage, website, 404, browser, page, 403, URL
J. Productivity Shortcuts	491 (1.23%)	17. Keyboard Shortcuts (491)	Focus: Editing/navigation efficiency in IDE/editor; Keywords: Ctrl, shortcut, comment, Shift, Tab, Enter, copy, indent
K. Data Visualization	427 (1.07%)	18. Charting & Mapping (427)	Focus: Plotting charts/maps and display settings; Keywords: map, color, display, country, province, bar-chart, CSV, GDP
L. Error Debugging	296 (0.74%)	19. Runtime Error Diagnosis (296)	Focus: Runtime error tracing and diagnosis strategies; Keywords: object, JSON, attribute, has, AttributeError, TypeError, NoneType

To further synthesize the findings, the twelve themes were consolidated into three categories based on their dominant support function (Table 4). Operational support needs (Themes B, D, F, H, J; 44.53%) encompassed environment setup, tool integration, encoding issues, package management, and productivity-related operations, all of which were oriented

toward establishing and maintaining a functional programming environment. Knowledge-construction needs (Themes C, E, G, I, K, L; 28.42%) covered conceptual understanding, data-centric tasks, file handling, web-related topics, visualization, and error debugging. Instructional-oriented needs (Theme A; 26.95%) centered on course feedback regarding pacing, explanation clarity, and resource completeness. Notably, operational support needs constituted the largest single category of discourse, exceeding both knowledge-construction needs and instructional-oriented needs when each was considered independently.

Table 4. Typology of Learner Needs Identified from Comment Themes.

Learner-Needs Category	Definition	Associated First-Level Themes	Share of Discourse
Instructional-Oriented Needs	Needs related to the organization, clarity, pacing, and completeness of instructional delivery and learning resources	A. Course-related Feedback	26.95%
Operational Support Needs	Needs related to establishing and maintaining a functional programming environment for participation in learning tasks	B. Technical Onboarding & Tool Integration; D. System & Environment Configuration; F. Encoding & Input Issues; H. Package Management; J. Productivity Shortcuts	44.53%
Knowledge-Construction Needs	Needs related to understanding programming concepts, applying knowledge, and solving task-related problems	C. Core Programming Concepts; E. Data-centric Applications; G. File Handling; I. Web-related Topics; K. Data Visualization; L. Error Debugging	28.42%

Table 5 presents representative original-language excerpts for the twelve themes, together with English translations and coding rationales. The excerpts were selected from the topic-modeling output based on two criteria: inclusion of high-frequency topic keywords and clear illustration of the theme's dominant support function.

Table 5. Representative Original-Language Excerpts for Each Theme.

Theme	Original Chinese Excerpt	English Translation	Coding Rationale
A. Course-related Feedback	我怎么感觉随难度的增加up讲的也越快了	"I feel like as the difficulty increases, the content creator speaks faster too."	Instructional pacing feedback
B. Technical Onboarding & Tool Integration	不懂就问，华为i5matebook14，轻薄本，能带起来不，下了python和vs code，小白一个，求大神解答	"Can a Huawei i5 MateBook 14 handle it? I downloaded Python and VS Code, total beginner, help please."	Tool selection and hardware readiness inquiry

Table 5. *Cont.*

Theme	Original Chinese Excerpt	English Translation	Coding Rationale
C. Core Programming Concepts	把print（count）放在最后，和while是对齐的，这么改试试呢	"Put print(count) at the end, aligned with while. Try changing it like this."	Guidance on indentation and loop structure
D. System & Environment Configuration	英雄哥哥，我下载了request模块，运行的时候报错，好像下载错误了环境，请问怎么更换环境啊	"I downloaded the request module, but it throws an error when running. I seem to have installed it in the wrong environment. How do I switch environments?"	Environment configuration and virtual environment issue
E. Data-centric Applications	问一次各位，就是主学数据分析的话，在编程方面，会pandas之类的模块浅显应用够不够呀?	"If I mainly study data analysis, is a basic understanding of modules like pandas enough?"	Data analysis skill-level inquiry
F. Encoding & Input Issues	小白提问为什么我的中文注释显示的是正方形的小框框呀？？	"Beginner question: why do my Chinese comments display as little square boxes?"	Character encoding display issue
G. File Handling	显示错误： FileNotFoundError:【Errno 2】No such file or directory: 'NEWS.txt'这是文件位置不对吗	"Error shown: FileNotFoundError: [Errno 2] No such file or directory: 'NEWS.txt'. Is the file location wrong?"	File path error in read operation
H. Package Management	为什么我scipy直接用pip install scipy安装不了么	"Why can't I install scipy directly with pip install scipy?"	Package installation failure
I. Web-related Topics	既然beautifulsoup都能提取网页的信息了，为何还用re正则式去提取呢？？？	"Since BeautifulSoup can already extract web page information, why still use regex for extraction???"	Web scraping technique comparison
J. Productivity Shortcuts	有大佬知道CTRL+左键无法进入类和对象怎么办嘛，一直提示connot find declaration to go to	"Does anyone know what to do when Ctrl+click can't navigate to classes and objects? It keeps saying cannot find declaration to go to."	IDE navigation shortcut troubleshooting
K. Data Visualization	老师好，请问子图绘制那一节，我程序和您写的一样，但是每个图像的开头最左侧都不在y轴上，都会缺一块，这是为什么?	"Teacher, in the subplot section, my code is the same as yours, but the left edge of each image does not align with the y-axis and part of it is missing. Why?"	Data visualization rendering issue
L. Error Debugging	一直出AttributeError: 'str' object has no attribute 'data'，不知道怎么转啊	"I keep getting AttributeError: 'str' object has no attribute 'data', and I don't know how to convert it."	Runtime error diagnosis

Note: English translations are provided for presentation purposes.

4.2. Mapping Learner Needs to the CoI Framework

4.2.1. Mapping the Learner-Needs Typology to the CoI Framework

Having established a data-driven typology of learner needs, we then examined how each category aligns with the CoI framework. As summarized in Table 6, the mapping revealed notable differences in explanatory coverage across the three categories. Instructional-oriented needs (Theme A, 26.95%) aligned closely with teaching presence, which emphasizes instructional design, facilitation, and direct instruction. Knowledge-construction needs (Themes C, E, G, I, K, L; totaling 28.42%) aligned with cognitive presence, reflecting diverse inquiry and meaning-making processes. However, operational support needs (Themes B, D, F, H, J; totaling 44.53%), which constituted the largest category, were not fully accounted for by any existing CoI presence.

Table 6. Alignment of Learner-Needs Typology with the CoI Framework.

Learner-Needs Category	Associated First-Level Themes	Dominant CoI Presence	Share of Discourse	Coverage Assessment
Instructional-Oriented Needs	A. Course-related Feedback	Teaching Presence	26.95%	Largely accounted for
Knowledge-Construction Needs	C. Core Programming Concepts; E. Data-centric Applications; G. File Handling; I. Web-related Topics; K. Data Visualization; L. Error Debugging	Cognitive Presence	28.42%	Largely accounted for
Operational Support Needs	B. Technical Onboarding & Tool Integration; D. System & Environment Configuration; F. Encoding & Input Issues; H. Package Management; J. Productivity Shortcuts	Not fully covered by existing presences	44.53%	Coverage gap identified

At the topic-modeling level, social presence did not emerge as a standalone learner-needs category. As brief social cues may be embedded within task-oriented comments, we further examined dominant and secondary presence labels through comment-level validation.

4.2.2. Comment-Level Validation Results

The comment-level validation provided evidence for assessing the boundary validity of the theme-level mapping at the individual-comment level. Cohen's kappa for dominant presence labels was $\kappa = 0.66$, with 80.33% observed agreement (241/300), indicating acceptable-to-substantial agreement for an exploratory validation involving theoretically adjacent categories [46]. In the consensus-coded sample, Technical-Operational was the most frequent dominant label (181/300, 60.3%), followed by Teaching (69/300, 23.0%), Cognitive (38/300, 12.7%), and Social (12/300, 4.0%).

Disagreements ($n = 59$) mainly arose when distinguishing Teaching from Technical-Operational (19 instances) and Teaching from Cognitive (17 instances), particularly when comments combined tool-use requests with instructional feedback or raised conceptual questions in troubleshooting contexts. Among the 125 comments sampled from operational support themes (B, D, F, H, and J), 88 comments (70.4%) were coded as dominant Technical-Operational, consistent with the interpretation that these themes primarily captured infrastructure and operational support needs. All discrepancies were resolved through discussion and joint review.

Secondary-presence coding further clarified the role of social presence. Social cues appeared in 61 of the 300 comments (20.33%). Only 12 comments (4.00%) were coded as dominant Social Presence, whereas 49 comments (16.33%) contained social cues only as secondary features embedded within task-focused discourse. These results indicate that social cues appeared primarily as secondary features rather than as the dominant communicative function.

4.2.3. Internal Structure of Operational Support Needs

To examine operational support needs in detail, we analyzed the five associated themes: Technical Onboarding & Tool Integration (Theme B), System & Environment Configuration (Theme D), Encoding & Input Issues (Theme F), Package Management (Theme H), and Productivity Shortcuts (Theme J). These discussions addressed operational prerequisites for learning activities. Comments frequently expressed inability to proceed (e.g., "cannot access the environment...cannot proceed with learning"), suggesting that technical issues were experienced as practical problems learners needed to address before continuing practice.

To understand the internal structure of operational support needs, we examined patterns within themes B, D, F, H, and J and identified three conceptual subdimensions: Infrastructure Configuration, Tool & Dependency Operation, and Technical Troubleshooting (Table 7).

Table 7. Types of Technical Support Needs in Learner Comments.

Dimension	Definition	Example Indicators	Primary Themes
Infrastructure Configuration	Community provision of support for installing and configuring development environments.	IDE installation, interpreter setup, PATH variables, version management	B , D
Tool & Dependency Operation	Community provision of guidance for operating development tools and managing software dependencies.	Package installation (pip/conda), virtual environments, module importing, IDE features	H , J
Technical Troubleshooting	Community provision of collaborative processes for diagnosing and resolving system-level failures.	Error diagnosis, encoding issues, compatibility problems, access permissions	F

4.2.4. Problem-Focused Learning Pattern

The analysis of comment content and posting timing reveals a distinctive learning pattern in this context: problem-focused engagement. In this mode, videos function primarily as searchable repositories rather than structured curricular content. Learners concentrate their interactions on specific issues. For example, a comment such as, "Why

wasn't the if statement used in the Day 7 exercise?" does not constitute a critique of the instruction but reflects an active comparison of alternative approaches. Another comment, "The code runs fine in the video, but throws an error on my computer, why?" illustrates how the comment section serves as a problem-solving space bridging abstract concepts and concrete implementation. This pattern reflects a just-in-time learning dynamic, where learners engage with the material in response to emerging problems rather than following a predetermined sequence.

Within this problem-focused process, operational readiness emerges as a critical prerequisite. Learners treat code execution as integral to the process of understanding, iteratively resolving issues through repeated testing and validation. Comments frequently indicate that learning is interrupted when environment setup or tool configuration fails. For instance, "The code won't run... stuck here" or "Spent two hours on pip install, can't move forward." These patterns indicate that, in this learning context, learners frequently described technical-infrastructure failures as interrupting their intended learning activities and as an immediate reason for pausing or being unable to continue practice.

5. Discussion

5.1. Technical Presence: A Context-Sensitive Extension to the CoI Framework

In our data, technical-infrastructure challenges formed a substantial and distinct category of learner needs in social-media-based programming education. At the theme level, 44.53% of discourse focused on environment configuration, dependency management, and tool integration. Learners consistently described these challenges as disrupting their engagement with instructional content, the testing of ideas through code execution, and collaboration with peers. The existing CoI framework, with its emphasis on instructional design, cognitive meaning-making, and social relationship-building, does not explicitly account for these operational prerequisites for practice-based learning. Viewed from a systems perspective, these patterns reveal a structural gap in how social media education systems support learners: the infrastructure layer that enables all other forms of engagement lacks systematic provision. Prior work reports similar patterns: setting up coding environments is often difficult, time-consuming, and confusing for students [47], motivating simplified tools such as MOCSIDE [48]. During the COVID-19 transition, Garcia et al. [49] showed that synchronous programming labs worked reliably when execution environments were centrally provisioned and standardized. In social-media contexts, however, learners work without unified infrastructure and must troubleshoot configuration and dependency issues on their own. Nguyen et al. [50] found that beginning with simplified, no-setup browser editors yields better early outcomes and preserves performance after transitioning to full IDEs, suggesting that technical infrastructure shapes learning trajectories. In our corpus, pedagogical and cognitive moves embedded within technical discussions were often associated with infrastructure-related disruptions to intended learning activity. Drawing on these patterns, we propose Technical Presence as a context-sensitive extension to the CoI framework. We define Technical Presence as the extent to which a learning community provides accessible infrastructure support and collaborative troubleshooting processes that enable participants to configure, operate, and maintain functional computing environments, thereby supporting operational readiness for practice-based learning. The comment-level validation supported the recognizability of the presence coding scheme at the individual-comment level: across 300 sampled comments, two independent coders achieved acceptable-to-substantial agreement on dominant presence labels ($\kappa = 0.66$).

These operational support needs are not fully specified by any single existing CoI presence or extension. Table 8 positions Technical Presence against the traditional CoI presences and major extensions in the current literature.

Table 8. Technical Presence in Relation to Existing CoI Presences and Extensions.

Construct	Core Focus	Level of Analysis	Relation to Operational Readiness
Teaching Presence [1,44]	Instructional design, facilitation, direct instruction	Pedagogical process	Focuses on instructional design and facilitation; operational readiness is not treated as a distinct analytic focus
Cognitive Presence [45]	Meaning-making through inquiry, integration, resolution	Epistemic process	Technical failures may disrupt inquiry cycles, though this relationship is not the primary focus
Social Presence [40,43]	Affective expression, group cohesion, interpersonal connection	Relational process	Captures affective and relational interaction; does not directly specify operational readiness for executable practice
Learning Presence [15,16]	Self-regulation, goal-setting, effort management	Metacognitive process	Addresses learner self-regulation and metacognitive strategies; does not directly address environmental operability as a community-level condition
Emotional Presence [17,18]	Affective states, satisfaction, emotional engagement	Affective process	Addresses emotional responses to learning; does not directly model operational readiness conditions
Technology Sub-dimensions [25]	Technology as a means of supporting teaching, cognitive, or social processes	Distributed across three presences	Distributes technology-related elements across existing presences; operational readiness is not isolated as a distinct support condition

As Table 8 shows, existing presences and extensions each target a specific dimension of the online learning process. The technology sub-dimensions proposed by Şen-Akbulut et al. [25] are most directly relevant to the present study. Their model distributes technology-related elements across the three traditional presences and treats technology as a means of supporting teaching, cognitive, or social processes. The present analysis foregrounds a different issue: the operational readiness of the computing environment as a condition for executable practice. In our data, technical obstacles were often framed by learners as conditions for practice. When learners reported non-operational computing environments, they described disruptions to instructional engagement, cognitive inquiry, and peer support.

We conceptualize Technical Presence as an infrastructural support condition that supports operational readiness and the enactment of teaching, cognitive, and social presences through executable practice. Programming presents significant barriers for novices [3]. Cognitive challenges may slow learning; technical failures can disrupt executable practice at a more fundamental level. Unlike approaches that distribute technology across existing

presences (Table 8), operational readiness in our data functioned as a condition for practice itself: when computing environments were non-operational, learners described disruptions associated with infrastructural conditions rather than pedagogical or conceptual difficulty. Teaching presence emphasizes instructional design and scaffolding, yet development tools, particularly IDEs, are often treated as passive elements even though they mediate programming activity itself [31]. Uysal [4] showed that inappropriate development environments impose additional cognitive load on programming tasks, and non-executable code can prevent learners from validating understanding through practice [51]. Infrastructure typically remains invisible when functioning properly but becomes salient at points of breakdown [52], a pattern reflected in our corpus through recurrent discussions of environment errors, dependency conflicts, and tool incompatibilities. From an activity-theoretic perspective, infrastructure operates as a mediating artifact between learners and practice [53]; tool failures can disrupt the activity system and interrupt goal-oriented action even when pedagogical design and cognitive readiness are in place. In the validation sample, social presence appeared as a secondary feature in technical discussions, with relationship-building rarely serving as the dominant communicative function.

Technical Presence therefore functions as an infrastructural support condition within tool-dependent, practice-based learning (Figure 5). When learners report environment failures, they describe being unable to run code, check intermediate ideas, or work through tasks; under these conditions, instructional design becomes difficult to enact, cognitive exploration is constrained, and collaboration is limited by incompatible setups. When infrastructure is stable, learners can more readily follow instructional materials, test concepts through practice, and coordinate problem-solving with others. Recognizing Technical Presence as a context-sensitive extension thus provides a clearer account of the substantial technical discourse observed in social-media-based programming education, one that traditional CoI categories do not fully specify.

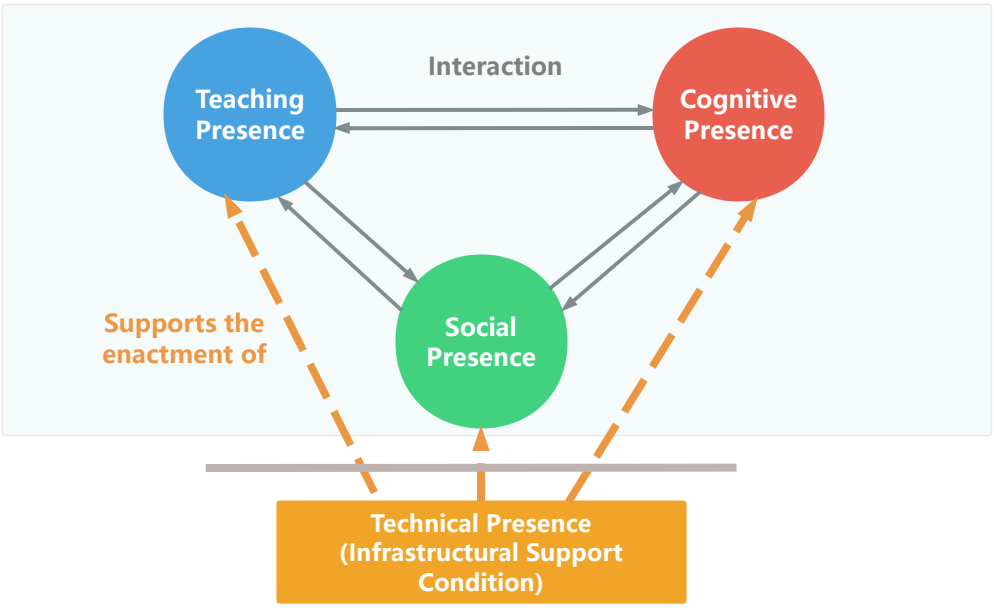


Figure 5. Technical Presence as a Context-Sensitive Extension to the CoI Framework in Tool-Dependent Learning.

5.2. Technical Presence as a Systemic Support Condition

Infrastructure support in programming education constitutes a systemic concern. Broader perspectives on collective learning and practice-based participation help clarify why.

The comment-section interactions in our data suggest an emergent form of collective learning: learners jointly diagnose configuration failures, share environment-specific solutions, and build a distributed pool of operational knowledge that extends beyond individual experience. Kimmerle et al.(2015) [54] characterize such dynamics in social media contexts as self-organized processes of collective knowledge construction, where individual learning and community knowledge co-evolve through bottom-up participation. This pattern is consistent with Senge's (1990) [55] broader argument that systems thinking and team learning are essential for developing shared understanding within any organized group. However, the prevalence of operational support needs (44.53%) suggests that this emergent collective learning process is shaped by infrastructure-level barriers that the community may not efficiently resolve through ad hoc comment threads alone.

The concept of communities of practice [56] offers a complementary lens. Programming tutorial comment sections can be understood as informal communities organized around a shared domain, where learners develop a shared repertoire of troubleshooting strategies through mutual engagement. Yet our analysis reveals a structural limitation that is especially salient in technology-mediated learning: when learners operate across fragmented combinations of operating systems, IDEs, and library versions, shared solutions become difficult to reproduce and transfer. In this sense, the infrastructure underlying practice becomes part of the learning problem itself. Wenger et al. (2009) [57] further argue that communities require deliberate technology stewardship to ensure that digital tools serve community goals. In the social media education ecosystem we studied, such stewardship is largely absent, leaving the community to address infrastructure challenges through ad hoc peer support in comment threads.

Technical Presence captures this systemic need. It refers to the extent to which a learning environment—whether formal or informal—ensures operational readiness through accessible infrastructure support and collaborative troubleshooting. Under this view, infrastructure is not peripheral to learning; it shapes the conditions under which instructional participation, cognitive inquiry, and the accumulation of shared practical knowledge can proceed.

5.3. Practical Implications

Recognizing Technical Presence as an infrastructural support condition in social-media-based programming education implies a shift in emphasis: instructors need to ensure that learners can reliably execute code while engaging with conceptual content. For content creators and instructors, this means treating executability as an explicit learning objective. Studies on configuration-free browser environments report that reducing local setup burdens can improve novices' early learning performance without harming later transition to full IDEs [50]. Accordingly, instructors can strengthen Technical Presence by adopting cloud-based editors and by providing standardized environment artifacts such as requirements files, conda environment specifications, Docker images, or preconfigured development containers. Structured debugging scaffolds can also help learners move systematically from encountering an error to diagnosing and resolving it.

For platform developers, our results point to the value of turning unstructured comment streams into usable troubleshooting resources. Issue-report templates and automated aggregation of high-frequency configuration problems can reduce redundancy and accelerate problem resolution. Tagging systems that distinguish operational issues from conceptual questions could further help learners locate relevant solutions and reduce the time spent navigating lengthy comment threads.

For education system designers, these findings highlight the value of incorporating operational readiness into the design of technology-enhanced learning environments.

Standardized execution contexts, such as containerized environments [33] or centrally provisioned lab setups [49], can reduce the variability of learner configurations and stabilize the conditions under which instructional and cognitive activities take place. Integrating Technical Presence into system-level design means treating infrastructure support as a core component of the learning ecosystem.

The growing integration of generative AI into programming education introduces additional considerations for Technical Presence. AI-powered tools may assist novices in interpreting error messages and suggesting fixes [14,35], potentially changing how learners approach infrastructure-related troubleshooting. At the same time, some studies have raised concerns that heavy reliance on AI-generated code may weaken diagnostic reasoning [58], suggesting that AI support should complement learners' own problem-solving processes. As AI tools become embedded in programming workflows, the operational readiness dimension may shift from manual environment configuration toward the ability to evaluate, verify, and adapt AI-generated solutions within functioning technical setups.

5.4. Limitations and Future Research

This study has several limitations that define the scope of its claims. First, the dataset comes from a single China-based social media platform (Bilibili) and centers on Python learning. Platform-specific features, such as comment threading, video format, and community norms, may shape how learner needs are expressed and addressed. Similarly, Python's toolchain characteristics (e.g., pip-based package management, virtual environments, IDE diversity) may produce a barrier profile that differs from languages with more streamlined setups (e.g., browser-based JavaScript) or more demanding compilation requirements (e.g., C++). The present study therefore supports the conceptual transferability of Technical Presence as an infrastructural support condition in tool-dependent learning, but the specific topic proportions and barrier distributions observed here should not be generalized to other platforms, languages, or cultural contexts without further investigation.

Second, the cross-sectional design captures learner needs as expressed at specific points in time and does not trace how technical barriers accumulate, evolve, or resolve across individual learning trajectories. The thematic and coding analyses identified patterns in learner discourse, but they do not establish causal relationships between technical barriers and learning outcomes; such effects would require longitudinal or experimental designs. The comment-level validation was based on a balanced stratified sample (25 comments per theme) designed to assess coding reliability and boundary validity; it was not designed to estimate corpus-level prevalence of each presence type.

Third, the analysis was conducted on Chinese-language comments. Although many of the identified barriers, such as environment configuration, dependency management, and tool integration, are likely to appear in other programming-learning contexts, language-specific discourse patterns and community interaction styles may influence how learner needs are articulated and categorized.

Future research could examine the cross-platform and cross-cultural transferability of Technical Presence, compare programming languages to map toolchain-specific barrier profiles, and incorporate longitudinal or experimental methods to clarify how technical support influences persistence and learning outcomes. As generative AI becomes a routine part of novices' workflows, future work might ask how learners verify AI-generated solutions, how AI-mediated troubleshooting alters the distribution of Technical Presence, and whether platforms will need to provide new forms of infrastructural support to maintain reliable, reproducible learning environments. Methodologically, the present analysis characterizes learner needs from a static collection of comments. Future work could complement this approach with graph-based and spatio-temporal methods that

model the relational and temporal structure of learner discourse. For example, graph attention networks combined with contrastive learning have been applied to interaction structures in social media data [59], and dynamic relational graph approaches have been used to capture evolving spatio-temporal dependencies in complex systems [60]. Adapting such methods to learner-needs data could help trace how support needs relate to one another and how they emerge and shift over the course of learning.

6. Conclusions

This study analyzed 40,004 comments from programming tutorial videos on Bilibili, a major Chinese social media platform (2016–April 2025). Using BERTopic, we identified twelve discussion themes (RQ1) and consolidated them into three learner-needs categories (RQ2): instructional-oriented needs (26.95%), operational support needs (44.53%), and knowledge-construction needs (28.42%). Mapping this typology onto the CoI framework (RQ3) showed that instructional-oriented needs aligned with teaching presence and knowledge-construction needs with cognitive presence, whereas operational support needs, the largest category, were not fully specified by the traditional CoI presences or by the major extensions reviewed in this study. Comment-level validation ($\kappa = 0.66$) supported the recognizability of the coding scheme, and secondary-presence coding showed that social cues appeared primarily as secondary features within task-focused discourse. Drawing on these findings, we propose Technical Presence as a context-sensitive extension to the CoI framework: an infrastructural support condition through which a learning community supports operational readiness for tool-dependent, practice-based learning. When learners reported environment failures, they described disruptions to instructional engagement, cognitive inquiry, and peer collaboration, suggesting that operational readiness shapes the conditions under which the three traditional presences can be enacted. These findings carry implications for content creators, platform developers, and education system designers seeking to strengthen the infrastructural foundations of technology-enhanced learning. Future research should examine the transferability and boundary conditions of Technical Presence across platforms and languages, and explore how AI-mediated troubleshooting may reshape the operational readiness dimension.

Supplementary Materials: The following supporting information can be downloaded at: https://www.mdpi.com/article/10.3390/systems14060685/s1, Dataset S1: De-identified comment dataset analyzed in this study.

Author Contributions: Conceptualization, Z.T. and C.K.L.; methodology, Z.T. and W.W.; software, Z.T.; validation, Z.T., K.L. and W.W.; formal analysis, Z.T. and W.W.; investigation, Z.T. and K.L.; resources, Z.T. and W.W.; data curation, Z.T.; writing—original draft preparation, Z.T.; writing—review and editing, W.W. and C.K.L.; visualization, Z.T.; supervision, C.K.L. and W.W.; project administration, C.K.L. and W.W.; funding acquisition, Z.T. and C.K.L. All authors have read and agreed to the published version of the manuscript.

Funding: This research was jointly funded by the Major Humanities and Social Sciences Research Projects in Zhejiang Higher Education Institutions (Grant No. 2024QN119), the Research Project Funded by Zhejiang Provincial Department of Education (Grant No. Y202351762), and the Zhejiang Provincial Zhonghua Vocational Education Research Project (Grant No. ZJCV2023C14). The APC was funded by the authors.

Institutional Review Board Statement: Ethical review and approval were waived for this study because it exclusively analyzed publicly accessible, anonymized user-generated comments collected via the platform's official API. No personal identifying information was collected or processed. The study did not involve any intervention or interaction with human subjects.

Informed Consent Statement: Not applicable. This study analyzed publicly available online comments. No personal identifying information was collected, and no interaction with participants occurred.

Data Availability Statement: The de-identified dataset supporting the findings of this study is available in the Supplementary Materials (Dataset S1). All direct personal identifiers (e.g., usernames, user IDs, and links) were removed prior to sharing.

Acknowledgments: The authors would like to thank Yi Yang for assistance with data collection and preliminary analysis. The authors have reviewed and edited the output and take full responsibility for the content of this publication.

Conflicts of Interest: The authors declare no conflicts of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript; or in the decision to publish the results.

References

1. Garrison, D.R.; Anderson, T.; Archer, W. Critical inquiry in a text-based environment: Computer conferencing in higher education. *Internet High. Educ.* **1999**, *2*, 87–105. [CrossRef]
2. Stenbom, S. A systematic review of the Community of Inquiry survey. *Internet High. Educ.* **2018**, *39*, 22–32. [CrossRef]
3. Kelleher, C.; Pausch, R. Lowering the barriers to programming: A taxonomy of programming environments and languages for novice programmers. *ACM Comput. Surv.* **2005**, *37*, 83–137. [CrossRef]
4. Uysal, M.P. Evaluation of learning environments for object-oriented programming: Measuring cognitive load with a novel measurement technique. *Interact. Learn. Environ.* **2016**, *24*, 1590–1609. [CrossRef]
5. Lim, K.K.; Lee, C.S. Sharing is learning: Using topic modelling to understand online comments shared by learners. In *Proceedings of the International Conference on Human-Computer Interaction, Virtual, 24–29 July 2021*; Springer International Publishing: Cham, Switzerland, 2021; pp. 91–101.
6. Alasmari, O.A.; Singer, J.; Bikanga Ada, M. Do current online coding tutorial systems address novice programmer difficulties? In *Proceedings of the 15th International Conference on Education Technology and Computers, Barcelona, Spain, 22–24 September 2023*; Association for Computing Machinery: New York, NY, USA , 2023; pp. 242–248.
7. Timoshenko, A.; Hauser, J.R. Identifying customer needs from user-generated content. *Mark. Sci.* **2019**, *38*, 1–20. [CrossRef]
8. Li, S.; Xie, Z.; Chiu, D.K.; Ho, K.K. Sentiment analysis and topic modeling regarding online classes on the Reddit Platform: Educators versus learners. *Appl. Sci.* **2023**, *13*, 2250. [CrossRef]
9. Jatain, D.; Niranjanamurthy, M.; Dayananda, P. A hybrid bio-inspired fuzzy feature selection approach for opinion mining of learner comments. *SN Comput. Sci.* **2024**, *5*, 135. [CrossRef]
10. Blei, D.M.; Ng, A.Y.; Jordan, M.I. Latent Dirichlet allocation. *J. Mach. Learn. Res.* **2003**, *3*, 993–1022.
11. Grootendorst, M. BERTopic: Neural topic modeling with a class-based TF-IDF procedure. *arXiv* **2022**, arXiv:2203.05794.
12. Khodeir, N.; Elghannam, F. Efficient topic identification for urgent MOOC Forum posts using BERTopic and traditional topic modeling techniques. *Educ. Inf. Technol.* **2025**, *30*, 5501–5527. [CrossRef]
13. Zankadi, H.; Idrissi, A.; Daoudi, N.; Hilal, I. Identifying learners' topical interests from social media content to enrich their course preferences in MOOCs using topic modeling and NLP techniques. *Educ. Inf. Technol.* **2023**, *28*, 5567–5584. [CrossRef]
14. Guo, P.J. Six opportunities for scientists and engineers to learn programming using AI tools such as ChatGPT. *Comput. Sci. Eng.* **2023**, *25*, 73–78. [CrossRef]
15. Shea, P.; Hayes, S.; Vickers, J.; Gozza-Cohen, M.; Uzuner, S.; Mehta, R.; Valchova, A.; Rangan, P. A re-examination of the community of inquiry framework: Social network and content analysis. *Internet High. Educ.* **2010**, *13*, 10–21. [CrossRef]
16. Shea, P.; Hayes, S.; Uzuner-Smith, S.; Gozza-Cohen, M.; Vickers, J.; Bidjerano, T. Reconceptualizing the community of inquiry framework: An exploratory analysis. *Internet High. Educ.* **2014**, *23*, 9–17. [CrossRef]
17. Cleveland-Innes, M.; Campbell, P. Emotional presence, learning, and the online learning environment. *Int. Rev. Res. Open Distrib. Learn.* **2012**, *13*, 269–292. [CrossRef]
18. Jiang, M.; Koo, K. Emotional presence in building an online learning community among non-traditional graduate students. *Online Learn.* **2020**, *24*, 93–111. [CrossRef]
19. Castellanos-Reyes, D. 20 years of the community of inquiry framework. *TechTrends* **2020**, *64*, 557–560. [CrossRef]
20. Yang, D.; Wang, S.; Zhao, L. Relationships between cognitive presence and students' learning outcomes in online higher education: A meta-analysis. *Distance Educ.* **2025**, *46*, 669–690. [CrossRef]
21. Zulu, F.Q.B. Enhancing the quality of online teaching and learning of a research module through the community of inquiry framework. *S. Afr. J. Educ.* **2024**, *44*, 2558 . [CrossRef]

22. Kovanović, V.; Joksimović, S.; Poquet, O.; Hennis, T.; Čukić, I.; De Vries, P.; Hatala, M.; Dawson, S.; Siemens, G.; Gašević, D. Exploring communities of inquiry in massive open online courses. *Comput. Educ.* **2018**, *119*, 44–58. [CrossRef]
23. Sun, Y.; Franklin, T.; Gao, F. Learning outside of classroom: Exploring the active part of an informal online English learning community in China. *Br. J. Educ. Technol.* **2017**, *48*, 57–70. [CrossRef]
24. Li, T.; Pollettini Marcos, L.; Huang, W.; Kenley, C.R.; Douglas, K.A.; Madsen, E.A.; Fentiman, A.W. Learning MBSE Online: A Tale of Two Professional Cohorts. *Systems* **2023**, *11*, 224. [CrossRef]
25. Şen-Akbulut, M.; Umutlu, D.; Arıkan, S. Extending the community of inquiry framework: Development and validation of technology sub-dimensions. *Int. Rev. Res. Open Distrib. Learn.* **2022**, *23*, 61–81. [CrossRef]
26. Veletsianos, G.; Navarrete, C. Online social networks as formal learning environments: Learner experiences and activities. *Int. Rev. Res. Open Distrib. Learn.* **2012**, *13*, 144–166. [CrossRef]
27. Deng, L.; Tavares, N.J. From Moodle to Facebook: Exploring students' motivation and experiences in online communities. *Comput. Educ.* **2013**, *68*, 167–176. [CrossRef]
28. Jurado, F.; Redondo, M.A.; Ortega, M. Using fuzzy logic applied to software metrics and test cases to assess programming assignments and give advice. *J. Netw. Comput. Appl.* **2012**, *35*, 695–712. [CrossRef]
29. Boustedt, J.; Eckerdal, A.; McCartney, R.; Moström, J.E.; Ratcliffe, M.; Sanders, K.; Zander, C. Threshold concepts in computer science: Do they exist and are they useful? *ACM SIGCSE Bull.* **2007**, *39*, 504–508. [CrossRef]
30. Yeomans, L.; Zschaler, S.; Coate, K. Transformative and troublesome? Students' and professional programmers' perspectives on difficult concepts in programming. *ACM Trans. Comput. Educ.* **2019**, *19*, 23 . [CrossRef]
31. Depradine, C.; Gay, G. Active participation of integrated development environments in the teaching of object-oriented programming. *Comput. Educ.* **2004**, *43*, 291–298. [CrossRef]
32. Noor, N.F.M.; Saad, A.; Ibrahim, A.B.; Noor, N.M. The acceptance of an educational integrated development environment to learn programming fundamentals. *Inf. Technol. Learn. Tools* **2023**, *93*, 135. [CrossRef]
33. Apahidean, L.; Nita, S. Containerized environments for computer engineering education. In *Proceedings of the EDULEARN25, Palma, Spain, 30 June–2 July 2025*; IATED: Valencia, Spain, 2025; pp. 3626–3636.
34. Prather, J.; Reeves, B.N.; Denny, P.; Becker, B.A.; Leinonen, J.; Luxton-Reilly, A.; Powell, G.; Finnie-Ansley, J.; Santos, E.A. "It's weird that it knows what I want": Usability and interactions with Copilot for novice programmers. *ACM Trans. Comput.-Hum. Interact.* **2023**, *31*, 4 . [CrossRef]
35. Haindl, P.; Weinberger, G. Students' experiences of using ChatGPT in an undergraduate programming course. *IEEE Access* **2024**, *12*, 43519–43529. [CrossRef]
36. Bilibili Inc. Announces Second Quarter 2025 Financial Results. Available online: https://tools.eurolandir.com/tools/PressReleases/GetPressRelease/?ID=7781920&lang=en-GB&companycode=services (accessed on 21 August 2025).
37. Chi, M.; Ma, H.; Li, Y.; Zhou, H. Factors influencing the communication effect of online videos for AI knowledge learning: A case study of AI learning content on Bilibili. In *Proceedings of the Wuhan International Conference on E-Business, Guangzhou, China, 6–8 June 2025*; Springer Nature: Cham, Switzerland, 2025; pp. 367–378.
38. Holman, L.; Stuart-Fox, D.; Hauser, C.E. The gender gap in science: How long until women are equally represented? *PLoS Biol.* **2018**, *16*, e2004956. [CrossRef]
39. Reimers, N.; Gurevych, I. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. *arXiv* **2019**, arXiv:1908.10084. [CrossRef]
40. Garrison, D.R.; Arbaugh, J.B. Researching the community of inquiry framework: Review, issues, and future directions. *Internet High. Educ.* **2007**, *10*, 157–172. [CrossRef]
41. Braun, V.; Clarke, V. Using thematic analysis in psychology. *Qual. Res. Psychol.* **2006**, *3*, 77–101. [CrossRef]
42. Saldaña, J. *The Coding Manual for Qualitative Researchers*, 4th ed.; SAGE Publications: London, UK, 2021.
43. Rourke, L.; Anderson, T.; Garrison, D.R.; Archer, W. Assessing social presence in asynchronous text-based computer conferencing. *J. Distance Educ.* **1999**, *14*, 50–71.
44. Anderson, T.; Rourke, L.; Garrison, D.R.; Archer, W. Assessing teaching presence in a computer conferencing context. *J. Asynchronous Learn. Netw.* **2001**, *5*, 1–17. [CrossRef]
45. Garrison, D.R.; Anderson, T.; Archer, W. Critical thinking, cognitive presence, and computer conferencing in distance education. *Am. J. Distance Educ.* **2001**, *15*, 7–23. [CrossRef]
46. Landis, J.R.; Koch, G.G. The measurement of observer agreement for categorical data. *Biometrics* **1977**, *33*, 159–174. [CrossRef]
47. James, T.; Magana, A.J. Evaluating self-paced computational notebooks vs. instructor-led online lectures for introductory computer programming. In Proceedings of the 2023 ASEE Annual Conference & Exposition, Baltimore, MD, USA, 25–28 June 2023.
48. Cazalas, J.; Barlow, M.; Cazalas, I.; Robinson, C. MOCSIDE: An open-source and scalable online IDE and auto-grader for computer science education. In *Proceedings of the 53rd ACM Technical Symposium on Computer Science Education V. 2, Providence, RI, USA, 2–5 March 2022*; ACM: New York, NY, USA , 2022; p. 1114.

49. Garcia, M.; Quiroga, J.; Ortin, F. An infrastructure to deliver synchronous remote programming labs. *IEEE Trans. Learn. Technol.* **2021**, *14*, 161–172. [CrossRef]
50. Nguyen, H.A.; Bogart, C.; Šavelka, J.; Zhang, A.; Sakr, M. Examining the trade-offs between simplified and realistic coding environments in an introductory Python programming class. In *Proceedings of the European Conference on Technology Enhanced Learning, Krems, Austria, 16–20 September 2024*; Springer Nature: Cham, Switzerland, 2024; pp. 315–329.
51. Koorsse, M.; Cilliers, C.; Calitz, A. Programming assistance tools to support the learning of IT programming in South African secondary schools. *Comput. Educ.* **2015**, *82*, 162–178. [CrossRef]
52. Star, S.L.; Ruhleder, K. Steps toward an ecology of infrastructure: Design and access for large information spaces. *Inf. Syst. Res.* **1996**, *7*, 111–134. [CrossRef]
53. Kuutti, K. Activity theory as a potential framework for human-computer interaction research. In *Context and Consciousness: Activity Theory and Human-Computer Interaction*; Nardi, B.A., Ed.; MIT Press: Cambridge, MA, USA, 1996; pp. 17–44.
54. Kimmerle, J.; Moskaliuk, J.; Oeberst, A.; Cress, U. Learning and collective knowledge construction with social media: A process-oriented perspective. *Educ. Psychol.* **2015**, *50*, 120–137. [CrossRef]
55. Senge, P. M. *The Fifth Discipline: The Art and Practice of the Learning Organization*; Doubleday: New York, NY, USA, 1990.
56. Wenger, E. *Communities of Practice: Learning, Meaning, and Identity*; Cambridge University Press: Cambridge, UK, 1999.
57. Wenger, E.; White, N.; Smith, J. D. *Digital Habitats: Stewarding Technology for Communities*; CPsquare: Boston, MA, USA, 2009.
58. Jošt, G.; Taneski, V.; Karakatič, S. The impact of large language models on programming education and student learning outcomes. *Appl. Sci.* **2024**, *14*, 4115. [CrossRef]
59. Hu, J.; Zhu, L.; Yang, M.; Tang, B.; Dai, S.; Zhang, C. GAAC: A Robust Misinformation Detection Framework via Graph Attention and Adaptive Contrastive Learning. *Knowl.-Based Syst.* **2026**, 115664. [CrossRef]
60. Hu, J.; Tang, B.; Zhu, L.; Li, Y.; Hu, J.; Yang, G. PDR-STGCN: An Enhanced STGCN with Multi-Scale Periodic Fusion and a Dynamic Relational Graph for Traffic Forecasting. *Systems* **2026**, *14*, 102. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.