

# Transforming Programming Videos into Personalized Tutoring Experiences: An LLM-Driven Cognitive Apprenticeship Approach

WENGXI LI, University of Michigan, USA

ROY PEA, Stanford University, USA

NICK HABER, Stanford University, USA

HARI SUBRAMONYAM, Stanford University, USA

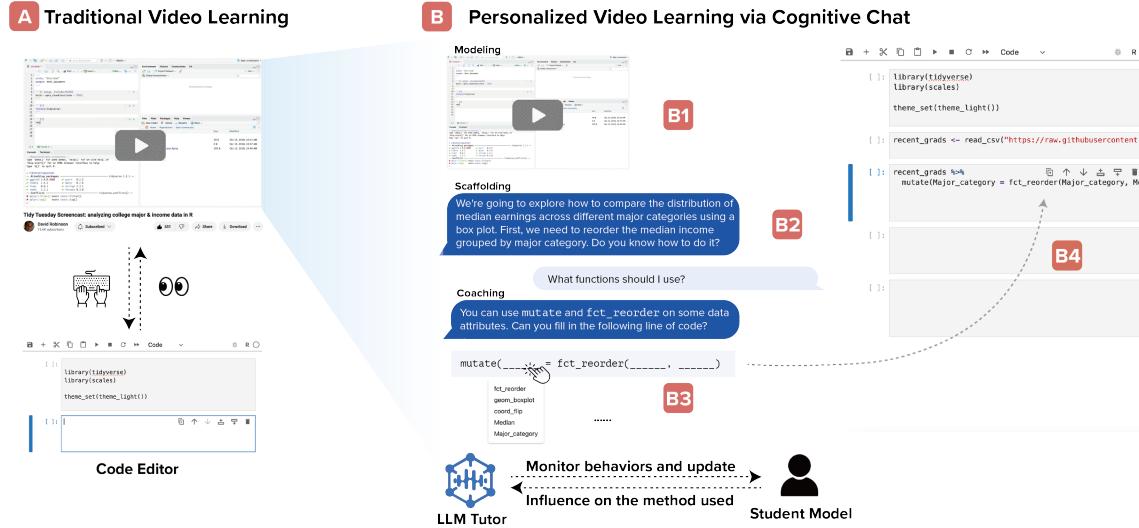


Fig. 1. Tutorly supports learners in personalized video learning via cognitive apprenticeship-based guidance. Compared with traditional video learning scenarios (A) in which learners switch between the video player and code editor to learn and practice, Tutorly (B) implements a generative AI tutor to transform video into video segments by learning goals (B1), uses different mentoring moves and monitors learner response (B2) to update the student model which can in turn adjust the mentoring moves (B3). In addition, Tutorly automatically syncs the code in the conversation to the Jupyter Lab notebook for learners to run (B4).

Online programming videos, including tutorials and streamcasts, are widely popular and contain a wealth of expert knowledge. However, effectively utilizing these resources to achieve targeted learning goals can be challenging. Unlike direct tutoring, video content lacks tailored guidance based on individual learning paces, personalized feedback, and interactive engagement necessary for support and monitoring. Our work transforms programming videos into one-on-one tutoring experiences using the cognitive apprenticeship framework. Tutorly, developed as a JupyterLab Plugin, allows learners to (1) set personalized learning goals, (2) engage in learning-by-doing through a conversational LLM-based mentor agent, (3) receive guidance and feedback based on a student model that steers the mentor moves. Based on a technical evaluation and a user study, Tutorly demonstrates a potential way of

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

© 2018 Association for Computing Machinery.

Manuscript submitted to ACM

53 generating structured and controlled guidance using LLMs and shows that proactive guidance and feedback in a cognitive manner are  
 54 advantageous to students.  
 55

56 **ACM Reference Format:**

57 Wengxi Li, Roy Pea, Nick Haber, and Hari Subramonyam. 2018. Transforming Programming Videos into Personalized Tutoring  
 58 Experiences: An LLM-Driven Cognitive Apprenticeship Approach. In *Woodstock '18: ACM Symposium on Neural Gaze Detection, June*  
 59 *03–05, 2018, Woodstock, NY*. ACM, New York, NY, USA, 38 pages. <https://doi.org/XXXXXXX.XXXXXXX>  
 60

61 **1 INTRODUCTION**  
 62

63 Video has emerged as a preferred medium for learning programming. According to a recent survey by Stack Overflow,  
 64 developers who learned through “How-to-videos” and “Video-based Online Courses” accounted for 60.14% and 49.41%,  
 65 respectively [55]. Video learning has several advantages, including convenient and asynchronous access [48], flexibility  
 66 to learn at different paces and focus on selective content [1], and several online content such as YouTube are free  
 67 or available at relatively low costs [24]. Regardless, the most effective learning paradigm for programming videos is  
 68 *learning by doing* [8]. However, this is easier said than done. Videos and supplement materials are often static and  
 69 spread out across different content formats and tooling (e.g., videos, slides, or GitHub repositories). Further, practicing  
 70 coding tasks while watching videos is difficult for novices, requiring clear instructions and immediate feedback [12].  
 71 Although students may occasionally interact with teachers who create video content (e.g., through online office hours),  
 72 this approach can be time-consuming and cumbersome[53].  
 73

74 In this work, we explore simulating instructor-guided practice using video content to enhance learning and support  
 75 students in effectively acquiring key concepts. As evidenced by research in AI and education [41, 61], Large language  
 76 models (LLMs) can be designed to facilitate learning programming by providing proactive guidance and real-time  
 77 feedback to students. However, operationalizing LLMs as tutors also requires addressing a number of current challenges.  
 78 First, LLMs often exhibit *verbosity*, favoring lengthy responses even when they are of lower quality and harder to  
 79 comprehend, despite prompts for brevity [17, 80]. Second, LLMs often struggle to *maintain focus* during multi-turn  
 80 conversations, leading to responses that go off-topic or deviate from the intended context, which can hinder consistent  
 81 and reliable guidance for learners [66, 69]. Third, effectively prompting LLMs to *teach students step-by-step* by monitoring  
 82 learning remains a significant challenge [68]. Effective tutoring requires summarizing key concepts, delivering concise  
 83 explanations, and guiding students through tasks based on their specific needs [49]. Consequently, LLMs become more  
 84 of a hindrance than a help for students [60].  
 85

86 To address these challenges in LLM-augmented learning and to empower learners to take an active role in learning,  
 87 we developed Tutorly, an LLM-powered *learning environment* for *interactive learning* from programming videos. Tutorly  
 88 is implemented as a JupyterLab <sup>1</sup> extension that supports conversational interactions for scaffolding and guidance in  
 89 learning from videos. We base our approach on the *Cognitive Apprenticeship (CogApp)* framework from the learning  
 90 sciences [19], which promotes six core moves: *Modeling, Coaching, Scaffolding, Articulation, Reflection, and Exploration*  
 91 (Figure 2). These methods help make both the instructor’s thought process and the student’s understanding visible.  
 92 For instance, a teacher might use MODELING to solve challenging math problems in class, while SCAFFOLDING  
 93 can include offering writing tips or physical support in activities like skiing. ARTICULATION encourages students  
 94 to verbalize their problem-solving strategies. The framework emphasizes tailoring teaching methods to a student’s  
 95 current skill level and selecting intermediate steps to guide their learning [57]. Using this framework, we implemented  
 96 a learning conversation generation pipeline that segments videos into distinct learning goals and, for each learning  
 97

98 <sup>1</sup><https://jupyter.org/>  
 99

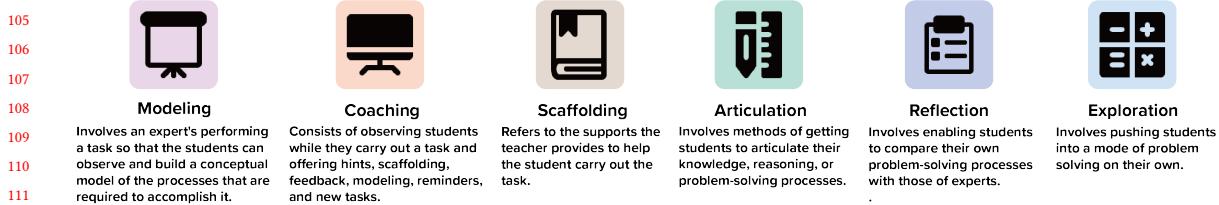


Fig. 2. The six cognitive apprenticeship methods.

goal, selects the appropriate teaching method from the *CogApp* framework based on an internal student model. As shown in Figure 1B, the system progressively generates conversational instructions for students to learn from video tutorials by applying different mentoring methods. This systematic process ensures a more controlled and relevant message generation tailored to the specific learning needs of students. Our approach ensures that even when enabling student-led Q&A interactions, Tutorly's instruction will not drift from the intended learning goals of the video.

We conducted a technical analysis of *CogApp prompting pipeline* and a within-subject user study of Tutorly's usability for helping students while watching videos. We found the guidance generated by our prompting pipeline is more concise and consistent across different learning goal and mentoring moves. Our approach for tightly integrating *CogApp* framework with LLMs *allows for targeted instruction* on knowledge from videos and *alleviates teaching constraints* through open-ended guidance generation. Based on our user study, participants demonstrated lower extraneous cognitive load and high germane load using Tutorly compared with traditional video learning. In summary, Tutorly highlights the potential of generative AI as a virtual instructor for video-based programming learning, combining pedagogy research on cognitive apprenticeship theory with research on the generative capabilities of LLMs.

Our key contributions include:

- (1) Tutorly: an **interactive conversational system** integrated into JupyterLab. Based on video content, Tutorly provides **student-specific** guidance that encourages practice aligned with learning goals.
- (2) *CogApp-based Prompting*: A **pipeline for generating learning conversations, using a domain-specific language** to organize *CogApp* methods. This enables LLMs to deliver guidance and interactions that are both **consistent and stable**, with options for personalized learning experiences.
- (3) **Evaluation** showing Tutorly's approach effectively generates structured and controlled instruction via LLMs, and users expressed a strong preference for using Tutorly to enhance their online learning experience.

## 2 RELATED WORK

The capabilities of LLMs offer novel perspectives to assist students in learning programming while watching tutorial videos. Here, we synthesize relevant literature on learning from videos, cognitive apprenticeship framework, and LLMs in intelligent tutoring systems (ITS), which is the basis for the design of Tutorly.

### 2.1 Theoretical Background

Learning to program requires integrating multiple skills, including understanding syntax, grasping programming concepts, and applying problem-solving strategies [39]. *Instructional Design Principles* guide how these learning experiences should be structured, including what content to present, how to sequence activities, and what teaching methods to use [22]. The *cognitive apprenticeship* framework is a widely used instantiation of these principles that build

on traditional apprenticeship models [19]. In this framework, the learner learns by observing, practicing under guidance, and gradually taking on more independence in the context of learning to program. Just as a *traditional* apprentice watches the expert demonstrate techniques before attempting them, CogApp makes expert programmers' thought processes and problem-solving strategies visible to novices. This gradual *transition of responsibility* helps manage learners' cognitive resources – breaking down complex programming tasks into manageable steps (reducing intrinsic load), providing clear instruction and support when needed (minimizing extraneous load), and encouraging productive practice and reflection as skills develop (fostering germane load) [71]. In other words, the framework emphasizes guided practice and self-regulation, where learners gradually take more control of their learning process. The knowledge taught through this apprenticeship model encompasses both *declarative* (factual information like programming concepts) and *procedural aspects* (step-by-step processes like implementing algorithms) [3]. Compared to other curriculum frameworks focusing solely on content delivery, the CogApp framework has shown effectiveness in both traditional classroom settings and informal learning environments by making expert thinking processes visible to novices through goal and sub-goal setting, explicit strategy instructions, and authentic tasks [6, 31, 35]. Our work implements the CogApp framework within a computational notebook environment where students interact with an LLM-powered tutor through rich conversational interactions that align with different mentoring moves.

## 2.2 Video-based Learning

Video-based learning leverages multimedia content to enhance student engagement and instructional delivery in programming and software development education [25, 58, 63]. While online platforms like MOOCs and YouTube offer scalable delivery, they often lack active engagement opportunities [16, 21, 30]. Studies emphasize that student control, epistemic agency, and creativity-boosting activities are important for effective learning [13, 32, 70, 81], with research showing that extensive practice and mistake-making improve knowledge retention [2, 46, 47, 67]. To address these issues, prior research has explored methods for enhancing video-based learning through interactive features and better design principles, such as embedding quizzes and annotations. For instance, Notestruct [40] and Videosticker [10] allow learners to create text-based and sketch-based annotations, enhancing their ability to interact with and reflect on video content. Quizzes embedded in videos support active learning by incorporating single- and multiple-choice questions [76, 77]. Navigation and segmentation tools further enhance video learning by identifying sub-topics or keywords to allow the learner to jump directly to relevant segments [23, 33, 34, 78, 79]. Recent advancements in LLMs also applied these techniques to transform video-based learning environments to offer personalized, context-sensitive support [18, 73]. Despite these advancements, current systems often lack the ability to dynamically segment video content based on learning goals and provide diverse, goal-aligned interactions tailored to each learner's needs.

Tutorly builds on these techniques while addressing key gaps in current video-based learning systems. First, Tutorly leverages LLM to segment videos based on learning goals. Unlike traditional systems that rely on complex segmentation algorithms or static annotations, Tutorly dynamically extracts knowledge from each video segment, ensuring that interactions are aligned with the intended learning goals. Second, Tutorly offers diverse interaction types, such as fill-in-the-blank questions and multiple-choice quizzes, surpassing tools that focus solely on annotations or quizzes. By integrating these capabilities into a JupyterLab plugin, Tutorly transforms passive video-based instruction into interactive, goal-driven tutoring experiences.

### 209    2.3 Conversational Intelligent Tutoring Systems

210    The evolution of Intelligent Tutoring Systems (ITS) began with Carbonell's Scholar system [11], which introduced a  
211    semantic representation of student knowledge to enable dialogue-based learning. Such an approach focuses on adapting  
212    to students, integrating domain-specific knowledge, and sharing control with learners [51]. A prevalent instantiation is  
213    conversational ITS (CITS), which facilitates natural language interactions to deliver real-time feedback and support,  
214    particularly in STEM education, where they assist with problem-solving through adaptive hints and dialogue [4, 52, 56].  
215    However, many existing CITS often rely on static domain knowledge and pre-scripted interactions, limiting their  
216    flexibility and scalability for personalized learning experiences. Recent advancements in LLMs offer the potential for ITS  
217    design by enabling context-sensitive, fluent, and adaptive dialogue with significantly reduced manual supervision [45].  
218    LLMs can now serve as tutoring agents across various domains, such as natural science [44, 68], language learning [42],  
219    computer science [27, 29, 64], and social skills coaching [28, 65]. These models enable more fluent, context-sensitive, and  
220    personalized dialogues, making it easier to adapt to students' diverse characteristics and needs [20, 43]. Despite these  
221    advancements, many current LLM-driven CITS do not fully integrate cognitive apprenticeship strategies or dynamic  
222    tracking of skill mastery, which are essential for guiding learners through complex skill development.

223    Tutorly incorporates a student module that tracks skill mastery dynamically, allowing it to select cognitive apprenticeship  
224    strategies tailored to each learner's current needs. By integrating LLMs with pedagogical principles, Tutorly  
225    overcomes the scalability and personalization challenges faced by earlier ITS. This approach offers an interactive,  
226    goal-aligned learning experience without extensive manual annotation, positioning Tutorly as a significant advancement  
227    in the field of conversational ITS.

## 233    3 USER EXPERIENCE

### 234    3.1 Interface

235    Tutorly is an integrated learning environment where users can watch programming video segments, engage in learning  
236    conversations with the AI mentor, and practice coding. Tutorly is implemented as an extension to *JupyterLab* – an  
237    interactive computational environment for notebooks, code, and data. The extension primarily consists of a rich *multi-*  
238    *modal* conversational chat panel added to the main JupyterLab interface. As shown in Figure 3, the default configuration  
239    consists of a notebook panel on the left and the chat panel (Tutorly) on the right, while users can adjust the position  
240    of the panels freely. In the notebook panel, the code is entered by learners or provided by Tutorly depending on the  
241    apprenticeship move. (Figure 3a). Instructors can use the “Settings” button (Figure 3c) to select specific learning goals  
242    and deeply engage with video segments corresponding to specific learning goals.

### 243    3.2 Apprenticeship Learning with Tutorly

244    To better understand how Tutorly transforms programming videos into a one-on-one mentoring experience, let us  
245    follow Leon, a student in a Data Science undergraduate program interested in learning EDA by watching tutorial videos.  
246    Leon's instructor recommends Tutorly and sets up his learning environment (see Section 3.2.3). Leon starts with the  
247    YouTube video “Tidy Tuesday Screencast: analyzing college major & income data in R <sup>2</sup>”.

248    3.2.1 *Understand the dataset.* To get the tutoring session started, Tutorly automatically loads up the dataset for Leon  
249    from the code repository corresponding to the video. Tutorly identifies that *understand the dataset* is the first relevant

250    

---

<sup>2</sup><https://www.youtube.com/watch?v=nx5yhXAQLxw>

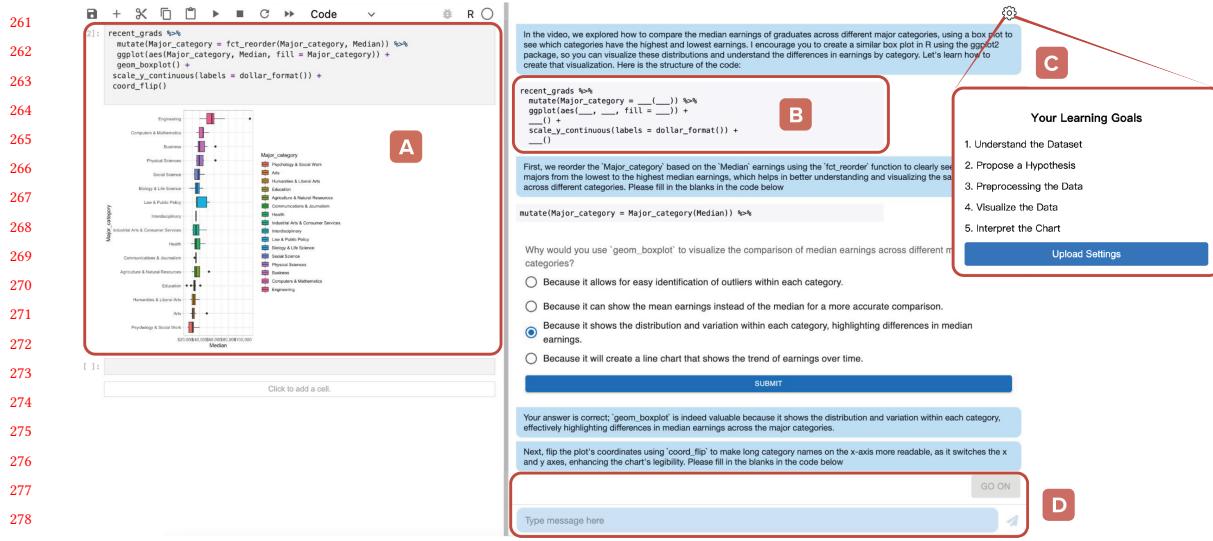


Fig. 3. Tutorly user interface as a JupyterLab Extension. Students can (A) write code in the cell-structured notebook, (B) practice under interactive guidance, (C) upload instructor-defined learning goals into the system, and (D) ask or answer questions, and proceed to the next video segment after completing all tasks

segment in the video. Tutorly takes the **MODELING** move and instructs Leon to watch the short video clip first. For goals involving *declarative knowledge* such as facts, **playing the video segment** in which the expert is exploring the dataset is a suitable first move for novices. After watching the video clip, Tutorly pops up a drop-down menu (Figure 4a), in which Leon can choose the attributes to see the meanings and statistics. Tutorly uses the **ARTICULATION** to let Leon illustrate his findings and hypothesis. Leon sends his answer to Tutorly and receives some feedback. The “Go On” button turns clickable, indicating that he is ready to proceed.

**3.2.2 Visualize the data.** The next step in the video is creating a visualization to explore the relationships between *College Major* and *Income*. Tutorly first uses **MODELING** to let Leon watch the video to learn the expert’s operation. Since this learning goal involves *procedural knowledge* by writing code to create a histogram, Tutorly uses the **SCAFFOLDING** to present Leon with the code snippet and walk him through the code lines (Figure 4b), and finally automatically adds the code to the notebook cell. Tutorly has access to the video’s completed code, which is used to generate the correct code. At the end of the **code walkthrough**, Tutorly uses **REFLECTION** to instruct Leon to execute the code and describe briefly what he has learned.

As the video progresses, Leon encounters a second instance of a box plot visualization. Because part of this process is similar to the previous histogram one, Tutorly takes a mix of the **SCAFFOLDING** and **COACHING** move and provides Leon with a **code template** for the box plot with key functions and parameters left blank (Figure 4c). The blanks in the code template are clickable, and Leon can select from the drop-down list of options. Based on the visualization intent, Tutorly guides Leon in filling out the blanks through conversation and interactivity. Tutorly provides feedback in case of incorrect selection (Figure 4d). Once completed, Tutorly adds the code to the notebook, which can be executed.

**3.2.3 Interpret the chart.** After Leon makes a visualization, Tutorly proceeds to the visualization interpretation part. Tutorly first uses **SCAFFOLDING** to provide specific guidance to Leon on what to attend to in the visualization (Figure 4e).

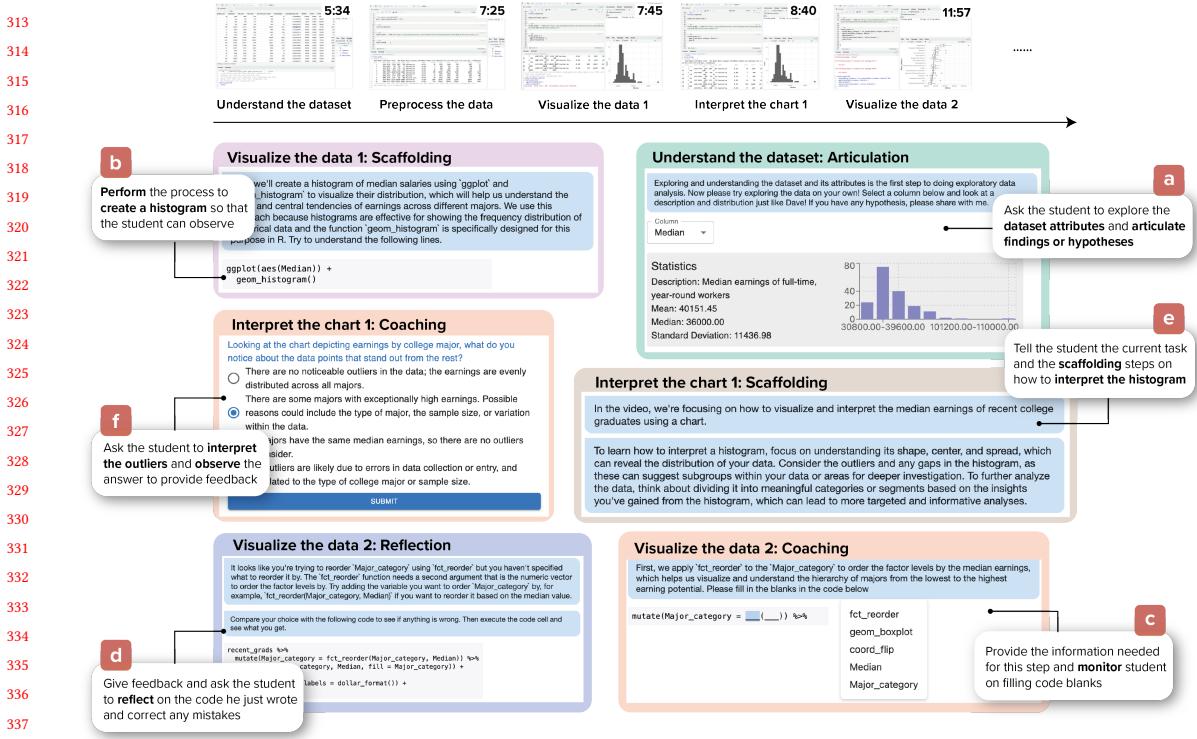


Fig. 4. Message examples using *CogApp* moves for different learning goals. For understanding the dataset, Tutorly uses (a) ARTICULATION. For visualizing the data, Tutorly uses (b) SCAFFOLDING, (c) COACHING, and (d) REFLECTION. For interpreting the chart, Tutorly uses (e) SCAFFOLDING and (f) COACHING.

And uses COACHING to give a multiple-choice questions for Leon to answer, such as asking Leon to interpret “the data points that stand out from the rest” (Figure 4f) as declarative knowledge. Then, Tutorly uses ARTICULATION to ask Leon to formulate hypotheses for the next visualization and provide feedback. Lastly, EXPLORATION allows Leon to go beyond what is contained in the source video, such as generating and testing hypotheses that are not covered in the video. Throughout these moves, the Tutorly constantly updates the underlying student model to capture the content and learning goals Leon has encountered.

In summary, Tutorly takes the source video, applies different mentoring moves from the *CogApp* framework, and interactively walks Leon through the video. In the learning conversations with Tutorly (Figure 5), Leon takes a learning-by-doing approach to develop his EDA and visualization skills and receives active feedback across different EDA tasks. In the process, Leon can also ask clarifying questions and work at his own pace and preference to regulate his learning.

### 3.3 Setting Up Tutorly: Instructor-Led and Self-Directed

Tutorly empowers instructors to define learning goals and customize pedagogical actions, ensuring alignment with curricular objectives and the diverse needs of learners. Using a web-based utility (Figure 6), instructors can articulate learning goals that specify what students should achieve after engaging with a video [15]. For instance, five goals

365  
366 After using MODELING to let the  
367 student watch the video segment,  
368 Tutorly turns to use SCAFFOLDING  
369 to briefly review the video content  
370 and introduce the current task

In the video, we explored how to compare the median earnings of graduates across different major categories, using a box plot to see which categories have the highest and lowest earnings. I encourage you to create a similar box plot in R using the ggplot2 package, so you can visualize these distributions and understand the differences in earnings by category. Let's learn how to create that visualization. Here is the structure of the code:

```
recent_grads %>%
  mutate(Major_category = ____) %>%
  ggplot(aes(____, ___, fill = ____)) +
  ____ +
  scale_y_continuous(labels = dollar_format()) +
  ____
```

371 Along with the previous message,  
372 Tutorly uses SCAFFOLDING to give  
373 student the code structure that has  
374 blanks, so that student can have an  
375 general idea of the task

376 Then Tutorly uses COACHING to let  
377 the student fill in the code line with  
378 blanks to finish the first line. If there  
379 is any error, Tutorly will provide hint

380 Then, we reorder the 'Major\_category' based on the 'Median' earnings using the 'fct\_reorder' function to clearly  
381 see the ranking of majors from the lowest to the highest median earnings, which helps in better understanding  
382 and visualizing the salary distribution across different categories. Please fill in the blanks in the code below

```
mutate(Major_category = Major_category(Median)) %>%
```

383 Why would you use `geom\_boxplot` to visualize the comparison of median earnings across  
384 different major categories?

Because it allows for easy identification of outliers within each category.  
 Because it can show the mean earnings instead of the median for a more accurate  
385 comparison.  
 Because it shows the distribution and variation within each category, highlighting differences  
386 in median earnings.  
 Because it will create a line chart that shows the trend of earnings over time.

387

The student has learned boxplot  
388 in the previous video segment, so  
Tutorly uses ARTICULATION to ask  
the student why use boxplot this  
time and provide feedback

389

390

391 Your answer is correct: 'geom\_boxplot' is indeed valuable because it shows the distribution and variation within  
392 each category, effectively highlighting differences in median earnings across the major categories.

393

Fig. 5. Details of messages in a conversation between Tutorly and a student who is learning how to compare the median earnings of graduates across major categories. According to the student model, Tutorly adopts different teaching methods to instruct different knowledge.

were established for Exploratory Data Analysis (EDA) videos [74]: loading datasets/packages, understanding datasets, preprocessing data, visualizing data, and interpreting charts. To enhance segmentation accuracy, instructors can include brief descriptions for each goal. For example, the goal *interpret the chart* might be described as “the video author interprets the visualizations and discusses the implications of the visualization, drawing conclusions, and theorizing about the underlying trends or patterns in data.”

Further, instructors can activate or deactivate specific learning goals for individual students, tailoring the experience to their needs (Figure 6a). Additionally, they can initialize the student model by inputting skill levels for each learning goal using Bayesian Knowledge Tracing (BKT) parameters [5]. This initialization personalizes guidance based on students’ prior knowledge and expected learning trajectories (Figure 6b). To further refine alignment with instructional strategies, instructors can preview the CogApp moves associated with video segments of each goal. They can also adjust actions and interactions for specific moves as needed (Figure 6c&d). Once satisfied, instructors can export these configurations as a JSON file, which students can upload into Tutorly to begin a customized learning experience.

Beyond instructor-led customization, Tutorly offers a self-directed mode where video creators can design topic-specific domain-specific languages (DSLs) to facilitate autonomous learning. These DSLs provide learners with a

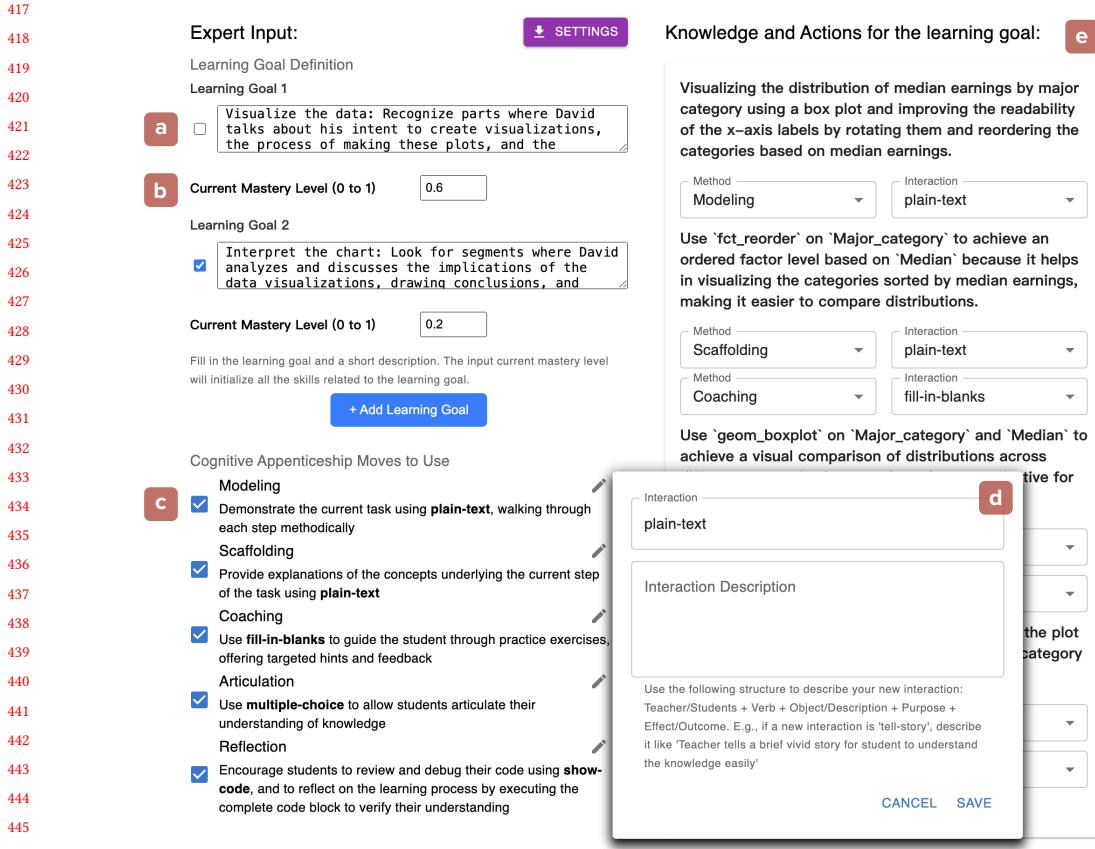


Fig. 6. Expert input interface. The expert can (a)define learning goals and choose to include them or not for the student, (b)initialize mastery level for each learning goal, (c-d)modify each move's action and interaction, (e)preview the pedagogy arrangement for an example video segment.

structured yet flexible framework to explore subject matter independently, enabling discovery and skill-building without direct instructor involvement. By tailoring DSLs to specific topics, creators ensure that self-directed learning remains purposeful and aligned with the content's pedagogical goals. Tutorly's hybrid design integrates both instructor-led customization and self-directed learning through DSLs, making it adaptable to a variety of educational contexts. This dual approach not only supports personalized, guided learning experiences but also fosters independent exploration, catering to diverse learning needs and preferences. An example DSL for EDA can be found in the Appendix.

## 4 SYSTEM ARCHITECTURE

Tutorly implements a set of LLM prompting techniques that yield conversational utterances grounded in the *CogApp* framework. As shown in Figure 7, our system consists of (1) a video segmentation module that *slices* long videos based on learning goals using a prompt chaining approach, (2) a DSL generator that produces the *context* for the LLM to generate utterances in interaction with the learner, and (3) a student model to capture the student’s learning progress

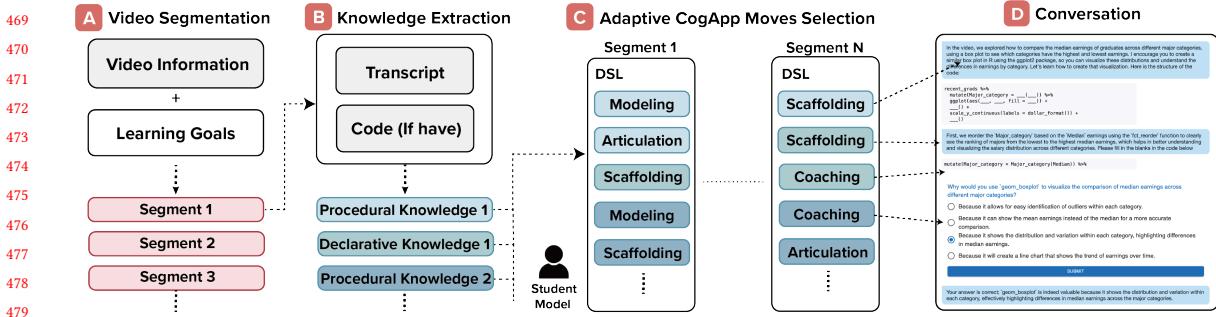


Fig. 7. Overview of *CogApp* pipeline: (A) the video transcript is segmented based on learning goals, (B) Tutorly extracts procedural and declarative knowledge from a video segment, (C) composes *CogApp* moves for all the knowledge identified in the video segment according to the student model and generates the DSL, (D) produces the interactive learning conversation based on the DSL.

and inform appropriate mentor *moves*. Here, we provide technical details about each of these modules along with challenges and prompt engineering strategies to make the generated conversations **consistent** and **controllable**.

#### 4.1 Video Segmentation by Learning Goals

When segmenting the transcript based on learning goals, small references to other goals (e.g., mentions of ‘data’ during a visualization segment) can cause GPT to misclassify segments. To mitigate this, we emphasize the contextual meaning of text chunks to minimize the influence of distracting words and phrases. Our approach is structured into three consecutive steps: (1) **Summarize**: Extract key points for each learning goal in the video, independent of timestamps; (2) **Retrieve**: Identify transcript sentences aligning with these summaries; and (3) **Rearrange**: Sequence the segments based on the start timestamps of the retrieved sentences. To ensure clarity, we provide few-shot examples of segmented videos, enabling GPT to generate distinct summaries for multiple occurrences of each learning goal within a video (see detailed prompt in Appendix A.2.1).

#### 4.2 Instructional Prompt Generator

The next step involves generating a teaching *plan* and associated *context* for each video segment. The plan needs to align with different moves in the *CogApp* framework to drive the conversational interactions with the learner later. In real life, if instructors were teaching, they would begin by determining specific knowledge or content that the goal should encompass [59]. Then, they would strategically determine pedagogical approaches for different learner needs. Similarly, our prompting strategy is to first use the segmented video transcripts and code to summarize the knowledge to be learned, then determine the teaching method based on the knowledge type for each of the moves in *CogApp* Framework. Finally, we map each method to predetermined actions and interaction types. By enforcing a step-by-step generation process, we emulate instructional practices and gain fine-grained control to carry out the conversation with learners progressively. Here, we detail each of these steps in generating instructional prompt templates:

**4.2.1 Summarize knowledge in each video segment.** In programming videos, declarative and procedural knowledge is spread across domain *concepts* (e.g., objects in object-oriented programming, dataset attributes) as well as *programming* constructs (visualization, sorting). We use few-shot prompts to summarize each video segment by generating concise procedural and declarative knowledge summaries. For each type of knowledge, we follow predefined summary formats

521 Table 1. Representations of the two types of knowledge in Tutorly. Each type of knowledge has concept-related and programming-related representations. The bold parts are later used in the student model.

522

524 Knowledge	525 Representation	526 Example
<i>Concept Related</i>		
527 Declarative	528 [Subject] + [verb phrase] + that + [independent clause].	529 <i>The median income by college major shows that majors earn a median income of over \$30K right out of college.</i>
<i>Programming Related</i>		
530 Procedural	531 To achieve/understand + [specific goal/outcome] + one must + <b>[actions/processes]</b> + [additional details] + considering/using + [relevant factors/tools].	532 <i>To understand the distribution of earnings by college major, one must <b>examine the histogram and identify overall trend or extreme values</b>, considering whether high earnings are due to the field's financial reward or influenced by factors such as low sample size and high variation.</i>
533 Declarative	534 The task is + [final goal] + using + [general method/tool] + and + [additional method/technique for enhancement].	535 <i>The task is comparing the distribution of median earnings across different major categories using a box plot and adjusting the visualization for better readability and interpretation.</i>
536 Procedural	537 To achieve + <b>[specific goal]</b> + one must + <b>[action/verb]</b> + <b>[specific tool/method]</b> + on + [object/target] + because + [reason/purpose].	538 <i>To achieve an ordered factor level based on the 'Median', one must use 'fct_reorder' on 'Major_category', making it easier to compare distributions.</i>

540  
541  
542  
543  
544  
545  
546  
547  
548  
549  
550  
551 (see Table 1). These formats help achieve three objectives. First, they enable the mapping of the summarized knowledge to the student model, as discussed in Section 4.3. Second, they ensure that the summaries are appropriately detailed for generating conversational messages with the LLM, avoiding excessively long or vague content. Third, the summary formats indicate the use of functions that are consistent with the video to avoid LLM generating different functions with the same functionality. For example, in Figure 8A, one piece of declarative knowledge and three pieces of procedural knowledge are extracted from the video segment, which contain the specific function names.

552  
553  
554  
555  
556  
557  
558  
559  
560  
561  
562  
563  
564  
565  
566  
567  
568  
569  
570  
571  
572 4.2.2 *Select Pedagogical Strategy for each piece of knowledge.* Using the knowledge summary as input, the next step is for the LLM to generate instructions grounded in the *CogApp* Framework. To ensure the apprenticeship moves are not selected randomly, our prompt engineering follows three guiding principles from the *CogApp* framework [19]: **global before local skills, increasing complexity, and increasing diversity.**

573 The first principle, *global before local skills*, emphasizes starting with a broad conceptual overview before addressing detailed tasks. Tutorly operationalizes this by leveraging LLM to dynamically analyze the knowledge summary and organize it into a global-to-local sequence. Based on this ordering, Tutorly begins with the **MODELING**, introducing the overall task and key concepts. Once the global framework is established, Tutorly transitions to **SCAFFOLDING** and **COACHING**, providing guided tasks that build on these high-level concepts. For example, in Figure 8C, Tutorly reorders the extracted knowledge to let the learner know the tasks in the video segment first.

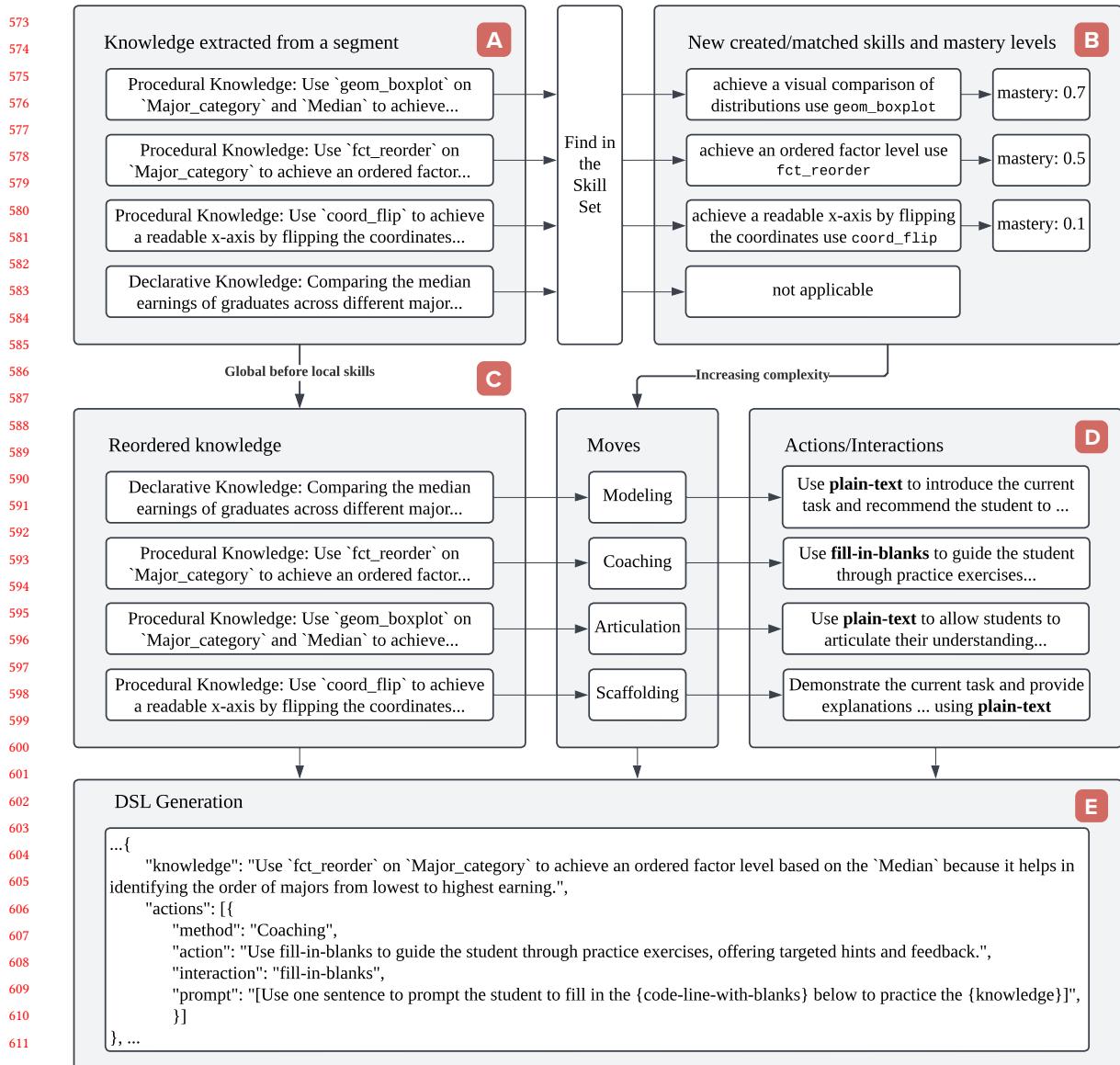


Fig. 8. Details of the instructional prompt generator: (A) Tutorly extracts procedural knowledge and declarative knowledge from the video segment, (B) the knowledge is mapped to skills and mastery levels, (C) Tutorly uses the principles to reorder the knowledge and decide moves for each knowledge, (D) moves are mapped to actions and interactions, (E) knowledge, moves, actions, interactions, and prompts are integrated to form the DSL.

The second principle, *increasing complexity*, ensures that tasks and instructional strategies adapt to the student's mastery of a skill, gradually increasing task difficulty and reducing reliance on guidance. At lower mastery levels, Tutorly uses **SCAFFOLDING** to provide detailed hints, step-by-step instructions, and illustrative examples. As the student demonstrates proficiency, these supports are gradually removed, fostering independent problem-solving and

critical thinking. The system also progressively increases task complexity. For example, an initial task may involve selecting the correct functions for a given task, while a more advanced task might require writing out an entire line of code. Additionally, if the mastery level for a skill exceeds a threshold (e.g., 0.7), Tutorly will have a greater probability that the practice of this knowledge is skipped (we set this probability equal to the mastery level of the skill).

The final principle, *increasing diversity*, addresses the need to maintain engagement and accommodate various learning contexts. Tutorly achieves this by varying task types and question formats. For instance, a task on interpreting a histogram might shift between multiple-choice question, open discussion, and annotation on the plot. Within the range of available interactions, we tailor the next interaction of each move, rather than letting the LLM randomly select.

**4.2.3 Generating a DSL for multi-turn conversation.** Directly inputting the knowledge summary and pedagogical move into the LLM is insufficient for generating coherent instructional dialogue. Teaching moves, as defined by the CogApp framework, are abstract principles. Without providing specificity and context, the LLM tends to produce responses that are overly verbose and unfocused, which diminishes learning effectiveness. Additionally, the conversation must closely align with the step-by-step progression of the video content. Traditional chatbots typically respond reactively to user input, but in the context of Tutorly, the conversation needs to proactively guide the student's learning process. This requires a controlled approach to ensure the dialogue remains on track, even when students' responses vary. Therefore, we need a reliable and controllable way to output messages.

To address these challenges, we developed a **domain-specific language (DSL) tailored for directing chatbot interactions in learning scenarios**. The DSL integrates the knowledge summary, pedagogical move, and context to produce well-structured messages. After identifying the pedagogical moves for each video segment, we use few-shot prompting to generate specific prompts for each move based on expert input that defines the expected action-interaction sequence. The *prompts* are stored in a queue. For example, in Figure 8D&E, each move has its corresponding action, interaction, and prompt.

The system follows a strict sequence of message generation. At each step, the head of the queue is used to produce the chatbot's next message. The student's response to the current message serves as an input parameter for the subsequent message. However, this response does not alter the overall order of messages or cause any messages to be skipped. This approach ensures that the conversation remains tightly focused on the video content and adheres to the intended sequence of CogApp moves.

### 4.3 Student Performance Monitoring

To adapt the conversation according to the student's performance, our approach models student performance by tracking skill proficiency through a Bayesian Knowledge Tracing (BKT) framework [5]. The first step in constructing the student model is to define the relevant student skills based on the knowledge outlined in the video segments. Each knowledge component is summarized using a standardized format (see Table 1), which encapsulates key information that translates directly into student skills. For instance, in Figure 8B, procedural knowledge in a programming-related video can be represented as “[specific goal] + [action/verb] + [specific tool/method]”, such as “achieve an ordered factor level use ‘fct\_reorder’”.

Once the student skills are defined, we initialize their proficiency levels using BKT parameters. These parameters estimate the probability that a student has mastered a given skill. At the start of a learning session, each skill is assigned an initial mastery level set by the instructor based on the student's overall proficiency for the current learning goal

(default to 0.1). For example, if an instructor estimates the initial mastery for the learning goal “Visualizing the data” to be 0.1, all this goal’s related BKT parameters are initialized accordingly.

As students interact with Tutorly—completing tasks, following instructions, and making errors—the BKT parameters are updated dynamically. This continuous update reflects the student’s evolving proficiency. When students encounter previously practiced knowledge, we use semantic similarity [50] to identify the relevant skills and ensure their BKT parameters are appropriately updated. This mechanism allows Tutorly to recognize when a student applies similar knowledge across different contexts and adjust the model accordingly.

At the end of each learning session, the updated BKT parameters are stored in a database. When a student begins a new learning module, Tutorly retrieves the stored parameters to initialize the student model, ensuring continuity in the learning experience. This approach enables Tutorly to personalize instruction by adapting to each student’s current knowledge state, maintaining coherence across different learning sessions.

#### 4.4 Implementation Details

Tutorly is implemented as a server-client architecture. The front end is a JupyterLab plugin<sup>3</sup> built using TypeScript and React. We make use of the provided APIs to integrate the plugin with the main computational notebook including @jupyterlab/notebook, @jupyterlab/cells, and @lumino/signaling so that the chat panel and the code cells in the notebook are linked and controlled by Tutorly. The chat UI design uses the Chat UI Kit from *chatscope* [14], an open-source UI toolkit for developing web chat applications based on React. The backend is built using Python and uses OpenAI’s API for all language model functions, specifically the GPT-4 model [54]. We use a temperature of 0.2 for mentioned prompts. We use the conversation buffer memory implemented by LangChain<sup>4</sup> to store messages during the chat. The video transcript and other necessary information are fetched using YouTube’s API. The original R Markdown code files and dataset files of videos are fetched using GitHub’s API. We use SQLite for the database design. We use the *Bilingual and Crosslingual Embedding for RAG* model [50] for text classification and semantic similarity. All server functions are implemented to ensure compatibility with the JupyterLab extension. The final open-sourced version can be installed on the JupyterLab desktop application in the extension shop.

## 5 TECHNICAL EVALUATION: VALIDATING COGNITIVE APPRENTICESHIP PROMPTING

To measure the segmentation accuracy, the controllability of generated messages using the *CogApp* prompting pipeline, and the importance of each component in the *CogApp* prompting pipeline across various instructional styles, we carefully selected three distinct video lesson topics: Exploratory Data Analysis (EDA), Machine Learning (ML), and Game Development. Each topic represents a unique approach to teaching. As shown in figure 9, EDA is characterized by a **cyclic structure** where data analysis is developed gradually through iterative steps, leading to **conclusions that emerge progressively** as the analysis unfolds. In contrast, ML takes a more **linear approach**, starting with a specific **practical task or problem**, then introducing **theoretical concepts** and principles, and finally reinforcing it by **programming** to solve the problem. Finally, Game Development lessons focus heavily on **demonstrations**, emphasizing creating and **visualizing game renders**, with less emphasis on drawing explicit conclusions. These choices were made to showcase Tutorly’s adaptability to different teaching methodologies, ensuring that it can effectively support various learning scenarios and content delivery styles. Concretely, we evaluate the *CogApp* prompting pipeline using three metrics:

<sup>3</sup><https://jupyter.org/>

<sup>4</sup><https://python.langchain.com/docs/modules/memory/types/buffer>

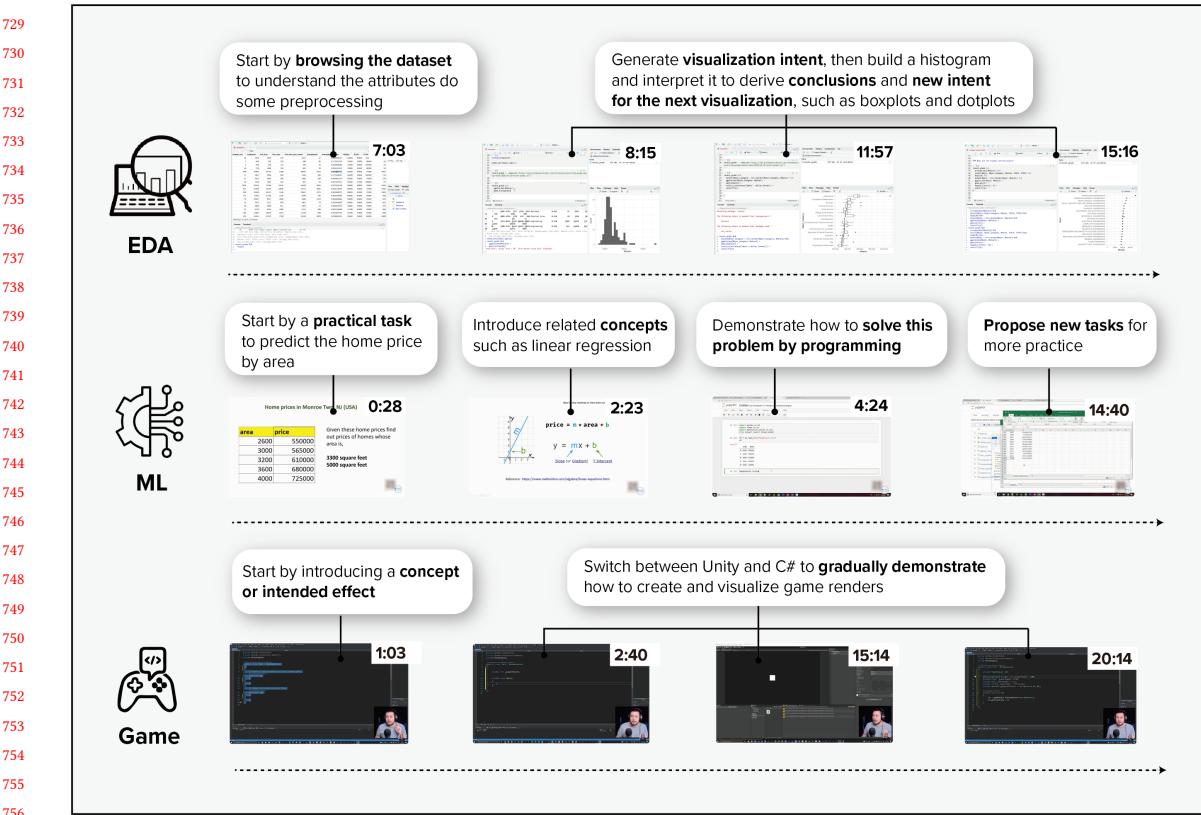


Fig. 9. Comparison of the three video lesson topics: Exploratory Data Analysis (EDA), Machine Learning (ML), and Game Development (Game). They have different proceeding structures.

- (1) **Segmentation Accuracy:** Prompting must correctly categorize the video transcript into segments corresponding to the learning goals.
- (2) **Controllability:** Experts can adjust the knowledge, methods, actions, and interactions in the prompting pipeline. Conversation must correctly follow the expert's settings to demonstrate the controllability of the pipeline.
- (3) **Component Importance:** Each component of the prompting pipeline is important and necessary to generate effective and reasonable guidance.

## 5.1 Procedure

The lead author and another expert, leveraging their expertise from reviewing numerous exploratory data analysis videos and familiarity with learning goals formulation, serve as the annotators for segmenting videos according to learning goals. Three additional experts recruited through our connections, proficient in each of the three video topics, served as evaluators to measure the accuracy of the intent.

*5.1.1 Evaluating Segmentation.* Two evaluators first watched one 20-minute video and segmented it individually. The percentage agreement within a 5-second tolerance threshold between evaluators was high ( $\rho = 0.88$ ). Then, one of the

Table 2. Prompt Pipeline Labels

Knowledge	Example
Procedural	To interpret the chart, one must analyze the histogram to identify outliers.
Declarative	Majors earn a median income of over \$30k right out of college.
Methods	Example
Modeling	I encourage you to follow along with me to create a meaningful visualization.
Coaching	Now, please fill in the blanks to add the box plot layer to our ggplot.
Scaffolding	You'll need to sort the data by median earnings in descending order.
Articulation	Can you tell me why it's important to consider these factors?
Reflection	Please compare your code with the standard code block.
Interactions	Example
plain-text	I'd like you to tell me what you observe about the high-earning majors.
multiple-choice	What could be the potential reason behind the pattern? Here are the options.
fill-in-blanks	Now, can you fill in the blanks to reorder the Major_category based on Median?
show-code	Please execute the cell to see the visualization.
annotation	Please use annotation to mark the specific areas on the chart.
Intent	Example
Task Control	We're going to focus on interpreting a chart related to median income by major.
Comprehension	Visualizing the distribution of median salaries helps us to see the spread of earnings across different majors, which can be very insightful.
Code/Run Code	The correct line should be recent_grads %>% ggplot(aes(Median)).
Feedback	You've done well in understanding the importance of each element in the plot!

students watched two more 20-minute videos and manually segmented them using the same learning objective definition as for GPT. We compared each segment’s timestamps between the labeled data and generated data. Considering that the transcript given to GPT is discrete and the information received by watching the video is continuous, we analyzed accuracy such that the timestamp differences are within five seconds error.

**5.1.2 Evaluating Controllability.** We created an initial set of dialog intent labels (Table 2) that have a hierarchical classification scheme. This scheme also represents the hierarchical prompt pipeline. The naming rules of labels are based on the previous Pair Programming Labels [62]. The ground truth data are extracted from the DSL files, which contains the details of all teaching methods defined by the expert instructor. The expert definition detail is consistent with the Table 2. The hierarchy begins with the root Knowledge classifier that categorizes knowledge into procedural and declarative knowledge. The second layer is teaching Methods defined in the cognitive apprenticeship theory [19]. We excluded *EXPLORATION* in the experiment because it would add open-ended dialogues that users could not control. In the third layer, we included all the interactions used to simulate conversations. Finally, we have four types of actions that represent the four most common behaviors in our programming teaching process [62].

Three expert evaluators label the data independently and separately for three topics of video (each topic has two videos): exploratory data analysis (124 utterances), machine learning (89 utterances), and game development (64 utterances). The expert chooses one label from each category to label the message so each message is annotated with four labels. For example, the evaluator would annotate this message: “Now, let’s delve deeper. I have a multiple-choice

833 question for you: What could be the potential reason behind the pattern of high earnings for certain majors? A) The  
834 field's inherent financial reward, B) Low sample size, C) High variation in the data, or D) All of the above." with the  
835 following label: "Declarative - Articulation - multiple-choice - Comprehension". Using this method, the evaluators  
836 annotated a total of 277 dialogue messages from the instructor side of Tutorly. Finally, we compared the labeled data  
837 with the ground truth data with the precision, recall, and F1-score metrics to evaluate the system's accuracy, providing  
838 an understanding of how well the system's output aligns with the expected results.  
839

840 5.1.3 *Component Importance.* To evaluate the component importance of *CogApp* prompting pipeline, we compare the  
841 expert assessment of *CogApp* prompt conflicts to ablated conditions. The first ablation is **(1) Baseline** unconstrained  
842 generation. In this ablation, we only consider the video segment information, ignoring teaching methods or knowledge.  
843 This case does not use *CogApp* theory and is closest to simply prompting a language model to teach. We also introduce  
844 a **(2) Knowledge-Only** ablation, where we consider only the knowledge induction step. This ablation summarizes  
845 knowledge of each segment but does not use *CogApp* theory as a backbone for method planning. In contrast, the **(3)**  
846 **Method-Only** ablation uses Cognitive Apprenticeship to generate messages but is not aware of the specific knowledge  
847 in the segment. Finally, we evaluate the entire *CogApp* prompting pipeline—knowledge induction and teaching method  
848 planning—and denote this as the **(4) Full** condition.  
849

850 For each ablation condition, we use LLM to simulate the conversations with each video segment's transcript and  
851 code (the prompt is in Appendix A.5). While blinded to which response came from which ablation, the evaluators  
852 ranked each set of four outputs (one from each condition) along three dimensions: credibility, validity, and interactivity,  
853 aggregating the scores to rank them from most to least ecologically valid. In all, we collected a total of 80 rankings  
854 (20 segments \* four conditions per dialogue), split evenly across two evaluators. Agreement between evaluators was  
855 moderate, with an averaged Spearman rank correlation of  $\rho = 0.71$ . After collecting the ranking data, we compute a  
856 TrueSkill score [26] for each ablation to quantify ablations' points, performed Kruskal-Wallis tests to determine overall  
857 differences between ablations, and applied Dunn's post hoc test [75] to isolate which specific condition differed from  
858 another.  
859

## 860 5.2 Results

861 5.2.1 *Tutorly has acceptable segmentation results.* Video transcript segmentation by learning goals algorithm achieved  
862 an overall accuracy of 76.9% within a five-second margin, which is an acceptable performance for educational  
863 content. However, the observed trend of decreasing accuracy with longer video duration suggests that the algorithm's  
864 effectiveness in identifying precise segmentation points diminishes over time. This observation may be attributed to  
865 the increasing complexity and variability of the content as the video progresses, making it challenging to maintain a  
866 consistent level of summarizing the video content and retrieving the relevant sentences. To alleviate this problem and  
867 improve segmentation accuracy, it could be helpful to split the video into smaller clips (e.g., every 10 to 12 minutes) and  
868 then segment by learning goals. This approach will limit the amount of content the algorithm needs to process at once,  
869 potentially reducing context load and allowing for more accurate identification of learning goals and corresponding  
870 segmentation points.  
871

872 5.2.2 *Tutorly is adept at interpreting user intent.* As shown in Table 3, the *Total* row provides an aggregate view of  
873 Tutorly's performance across all video topics while the other rows contain metrics of each video topic. The Knowledge  
874 column reflects Tutorly's capacity to extract procedural and declarative knowledge from the videos. This is a critical  
875 first step as it sets the foundation for subsequent instructional design. Precision and recall scores for different video  
876

Table 3. Comparison of the intent performance across three video topics: exploratory data analysis (EDA), machine learning (ML), and game development (Game) using the precision, recall, and F1-score metrics.

Topic	Knowledge			Method			Action			Interaction		
	Prec.	Recall	F1	Prec.	Recall	F1	Prec.	Recall	F1	Prec.	Recall	F1
Total	0.791	0.787	0.789	0.814	0.809	0.807	0.902	0.895	0.896	0.970	0.968	0.968
EDA	0.810	0.806	0.808	0.849	0.815	0.818	0.896	0.871	0.872	0.984	0.984	0.984
ML	0.795	0.798	0.796	0.825	0.809	0.807	0.900	0.899	0.899	0.956	0.944	0.947
Game	0.827	0.781	0.792	0.808	0.813	0.798	0.954	0.953	0.953	0.973	0.969	0.969

topics are around 0.791 and 0.787, indicating that Tutorly is relatively accurate in generating messages according to the given knowledge, without containing misleading information (high precision), and without missing important content (high recall). The slightly lower performance (recall equals 0.781) in game development videos may suggest the presence of more diverse or ambiguous content in such videos, which poses a greater challenge for knowledge extraction.

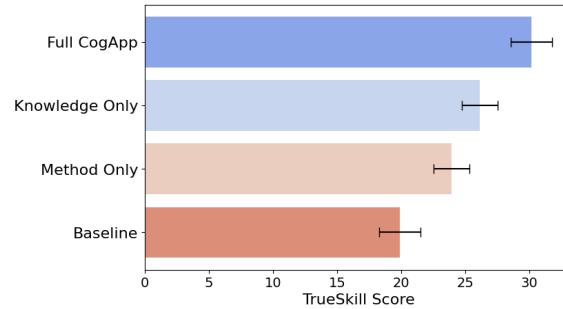
The *Method* column represents Tutorly's ability to generate information that is well aligned with a defined pedagogical move. Precision and recall are also consistent for this category (the difference between the highest and lowest recall is 0.006), with EDA videos showing better performance. This higher score suggests that Tutorly is more effective at following the given methods to generate appropriate messages in an EDA environment, which may be due to the inherently interactive and action-oriented characteristics of EDA tasks. The *Action* column metrics reflect how Tutorly translates interactions into specific messages or commands. This step also shows high accuracy (over 0.900), indicating that Tutorly can reliably trigger intended actions based on the user's setting. The higher scores in game development (0.954) suggest Tutorly's action messaging is particularly effective at identifying factual content and responding to action-driven commands in game scenarios, which may involve a mix of dynamic and static content requiring precise system responses.

In the *Interaction* column, the high scores (all around 0.950) signify Tutorly's proficiency in generating messages based on the selected interactions. This step is crucial as it determines how students will engage with the material and how users can personalize interactions based on their needs. The near-perfect performance on all three topics reflects Tutorly's robustness in seamlessly converting specified interactions into actual conversational utterances.

Along the process of Knowledge-Method-Action-Interaction, the scores show an upward trend (e.g., precision is 0.791-0.814-0.902-0.970 respectively). The ascending scores across these stages can be understood as a natural outcome of the instructional guidance generation process, which moves from the broad and general to the specific and interactive. At each step, Tutorly leverages the work done in the previous phase, refining and sharpening its outputs.

**5.2.3 The full CogApp prompting pipeline outperforms all ablations.** In Figure 10, we observe that the full condition has the highest TrueSkill score ( $\mu = 30.14, \sigma = 1.60$ ), suggesting that combining both knowledge induction and method planning is the most effective strategy. Knowledge-only condition ( $\mu = 26.11, \sigma = 1.40$ ) has a higher score than the method-only condition ( $\mu = 23.89, \sigma = 1.40$ ). This suggests that without knowledge induction, the method-only approach may fail to provide precise guidance tailored to the video content, potentially leading to over-scaffolding simple concepts or under-scaffolding more complex ones. In contrast, while the knowledge-only condition has a higher score than method-only approach, it lacks the adaptive capability provided by method planning, making it less responsive to the student's mastery level. The baseline condition has the lowest performance ( $\mu = 19.86, \sigma = 1.60$ ), confirming that unconstrained generation without any structured guidance is the least effective. Without restrictions, the conversations generated by ChatGPT are more likely to deviate from the teaching direction and reduce educational coherence.

937 Results from the Kruskal-Wallis test and the Dunn post-hoc test also reflect improvements provided by *CogApp*  
 938 prompting. The Kruskal-Wallis test indicates significant differences between conditions ( $p < 0.001; H = 32.86$ ).  
 939 Furthermore, the Dunn post-hoc test indicates significant pairwise differences between all ablations ( $p < 0.05$ ).  
 940 Altogether, these results confirm that all components in the *CogApp* pipeline are essential for generating more effective  
 941 and coherent guidance, thereby maximizing the educational potential of Tutorly.  
 942



943  
 944 Fig. 10. TrueSkill scores across ablations of the full *CogApp* prompting pipeline. The full prompting pipeline handily outperforms all  
 945 other conditions.  
 946

## 947 6 USER STUDY: MEASURING USER ACCEPTANCE AND COGNITIVE LOAD OF TUTORLY

948 We conducted a within-subject user study to investigate the usability and cognitive load of Tutorly. We recruited  $N = 16$   
 949 participants through our personal networks and social media messaging. The individual sessions were conducted via  
 950 Zoom and lasted approximately 2 hours. We compensated participants with \$50 for the time. At the start of the session,  
 951 participants were given instructions to install the extension on their computers or remotely control Tutorly deployed  
 952 on the lead author's laptop. Most participants opted for the latter as they also had to install JupyterLab, pip, and R on  
 953 their machines, which can be time-consuming.  
 954

### 955 6.1 Method

956 In each session, participants were asked first to fill in a pre-test questionnaire to assess their prior knowledge of EDA  
 957 topics. Next, participants were randomly assigned to one of two conditions: (1) watch the video directly on YouTube with  
 958 JupyterLab by the side, and ChatGPT with video-specific information, or (2) watch the video in Tutorly environment for  
 959 45 mins. Each condition was assigned a different video from YouTube <sup>5</sup> <sup>6</sup>. The sequence of conditions was randomized  
 960 across participants. For the Tutorly condition, participants also received a 15-minute tutorial on the main features and  
 961 could try out the system on their own before proceeding to the test tasks. In the second half of the study, participants  
 962 worked on the other condition, allowing for comparative feedback between watching videos with and without Tutorly.  
 963

964 Finally, participants were given a post-test quiz to fill in, in which there were five questions to test their knowledge  
 965 of the same learning goals of EDA after learning. Each participant then completed a usability questionnaire [37], a  
 966 cognitive load questionnaire [36], and a 10-minute open-ended discussion on the comparison of learning experience  
 967 between using Tutorly and not using Tutorly. The comparison are guided in three aspects: the influence of  
 968

969 <sup>5</sup><https://youtu.be/nx5yhXAQLxw?si=g6YmEUDeL3-vZ3Cw>

970 <sup>6</sup><https://www.youtube.com/live/Kd9BNl6QMmQ?si=JqXX5AbZ3r2fnLiG>

adding exercises while watching, what burdens are reduced or increased, and how to get a better experience. Two researchers coded 25% of the data independently and discussed their interpretations to reach a consensus. Then one of them coded the rest of the data. The final scheme included categories such as engagement enhancement, burden reduction, clarity of explanations, and personalization of learning experiences. The participant demographics are provided in the appendix.

## 6.2 Results

**6.2.1 Practice with Tutorly shows higher engagement in watching videos.** Across all sessions, participants responded positively to using Tutorly to learn programming through watching videos. Although 75% of the participants expressed that they typically practice programming while watching videos (i.e., learning by doing), more than half felt there was not enough explanation of the code or how to debug it (66.7%), and there were problems applying the concepts in practice due to a lack of detailed examples (58.3%). By contrast, as shown in Figure 11, 87.5% agreed or strongly agreed that they like using Tutorly's interface (Q10), and 81.25% agreed or strongly agreed that it was easy to learn to use the system (Q9). According to P2, when watching a long instructional video, the hardest thing is to stay focused. Instead, using Tutorly encourages them to become more immersed in watching videos and practicing: "*if I want to answer the questions or fill in the code blanks correctly, I should watch the video clip carefully to avoid multiple viewings.*"(P2) Similarly, P16 commented: "*I feel like I can actually practice instead of just watching videos numbly.*" P7 remarked: "*With Tutorly, I find myself paying more attention to the video content because I know I'll need to apply it right away. It's like having a tutor who's going to quiz me after each lesson.*" This immediate requirement to apply the material effectively addresses the common shortcoming of traditional video learning methods, which often lack sufficient detailed, practical examples.

**6.2.2 Tutorly demonstrates a better alternative to ChatGPT.** Based on feedback from using both Tutorly and baseline, although using ChatGPT to practice is quite common among participants (93.8%), over half (68.8%) point out it is hard to express their needs clearly, which led to frustration. In contrast, 93.75% of participants believed that the information provided by Tutorly effectively helped them complete tasks and scenarios (Q18), and 68.75% of participants thought that the information provided by Tutorly was clear (Q15). This high level of effectiveness suggests that Tutorly not only encourages careful video watching, as noted by P2, but also presents information in a more accessible and context-appropriate manner. For example, P10 commented that Tutorly does not generate long and tedious messages, which means it is easier to understand: "[*Tutorly*] doesn't give super long messages like ChatGPT, but it also has enough stuff so I won't get lost in it." P12 noted: "*With Tutorly, I don't have to worry about formulating the perfect query. The relevant information is right there, tied to the video I'm watching.*" Besides, P3 found it very convenient not to describe the video content to Tutorly: "*I always need to describe a specific situation before asking how to do it [with ChatGPT].*" This suggests that Tutorly offers a more personalized and engaging user experience compared to the often overwhelming responses from ChatGPT. It highlights the need for developing a tool that can deliver video-specific guidance, as effectively as Tutorly does.

**6.2.3 Tutorly reveals the potential for personalized learning.** Among participants who do not practice while watching, a common reason is "Uncertain about how to start or what to practice". 81.25% of participants agreed or strongly agreed that it was easy to learn to use the system (Q9), indicating its accessibility to learners with varied backgrounds. This ease of use, combined with the flexibility to configure learning goals and teaching methods, addresses the diverse preferences expressed by participants like P8, P14, and P9. According to P8: "*it's better if I can choose learning goals I wanna learn.*"

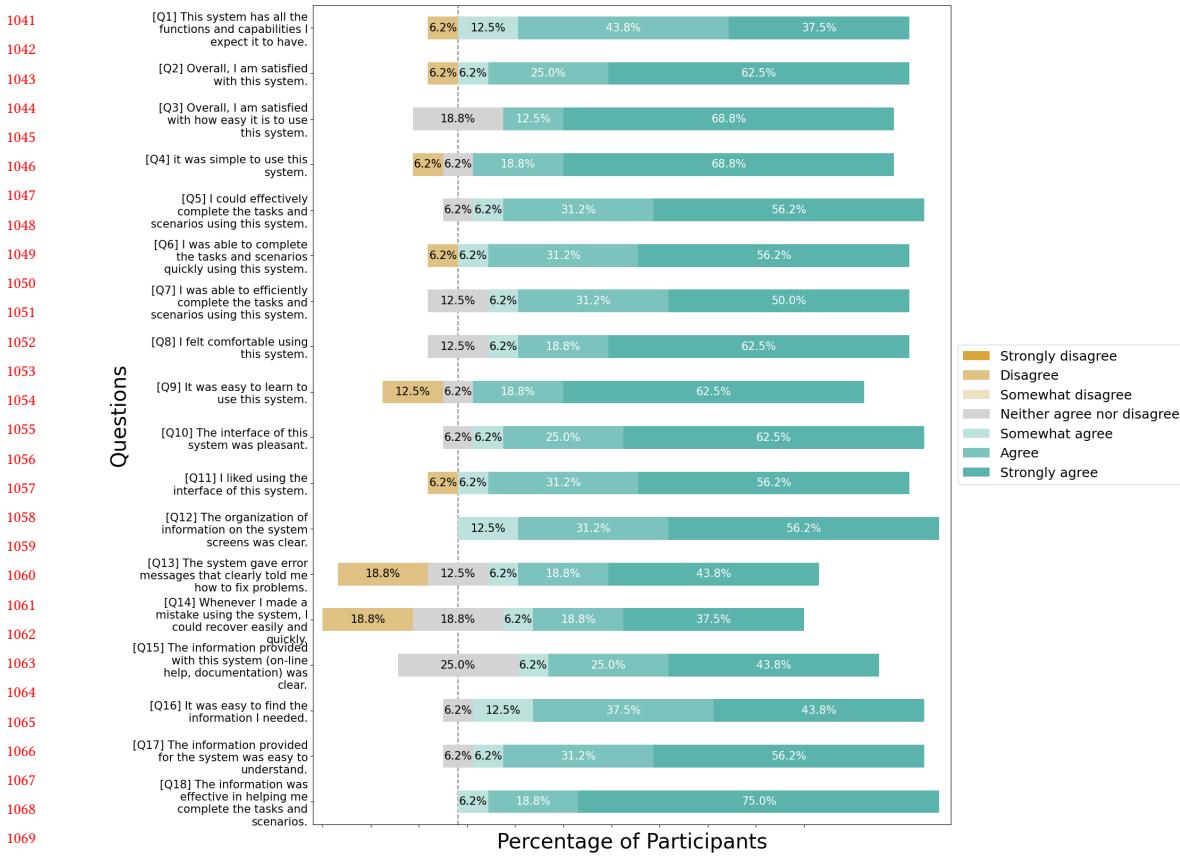


Fig. 11. Participant Responses to Post-Study System Usability Questionnaire.

Another participant (P14) pointed out, “*I dislike reading texts about code*”. P9 said, “*I prefer writing code than answering questions*.” Using our action configurator, we can flexibly support these learner preferences. This adaptability not only caters to individual learning styles but also potentially increases motivation and engagement over time. Participants also proposed additional features that could be added to make Tutorly more interactive, for example, P16: “*I appreciate how I can customize my learning experience. Some days I feel like coding, other days I prefer multiple-choice questions. It would be better if Tutorly supports more practice methods and lets me choose what works best for me at that moment*.”

**6.2.4 Overall usability of Tutorly’s interface.** In general, participants responded positively to the learning process with Tutorly. In the PSSUQ survey, we anchored at the endpoints with the terms “Strongly agree” for 7 and “Strongly disagree” for 1. We found the overall usability score for the system was 6.16 ( $SD = 1.27$ ). This aligns with the qualitative feedback, where the majority of users expressed positive sentiments about the interface and its ease of use. In terms of sub-scales, the System Usefulness (SYSUSE) score was 6.25 ( $SD = 1.23$ ), indicating that the system was perceived as relatively easy to use and learn by the respondents. However, the Information Quality (INFOQUAL) score of 5.95 ( $SD = 1.43$ ) points to some room for improvement in the clarity and comprehensiveness of the information provided. This echoes the participant feedback about the need for more detailed code explanations and debugging assistance.

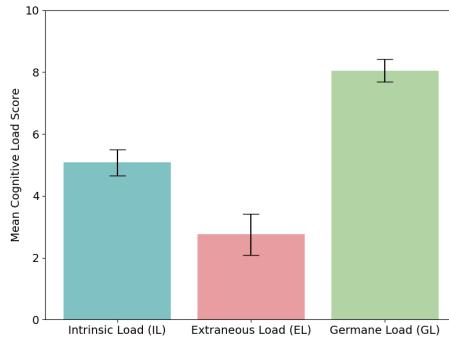


Fig. 12. Cognitive Load Results of Intrinsic Load (IL), Extraneous Load (EL), and Germane Load (GL).

We will talk about this in more detail in the next section. Encouragingly, the Interface Quality (INTERQUAL) score of 6.28 ( $SD = 1.06$ ) indicates that participants were quite satisfied with the visual appeal and overall experience of using Tutorly's interface. This positive perception of the interface design aligns with the high levels of user engagement and satisfaction as discussed earlier. These quantitative usability metrics, combined with the qualitative feedback, provide a well-rounded assessment of Tutorly's strengths and potential areas for improvement.

**6.2.5 Cognitive load of Tutorly's interface.** Tutorly presented a moderate level of intrinsic load, minimized extraneous cognitive demands, and maximized germane cognitive engagement for learning. In the cognitive load survey, we use a 10-point scale where 0 represents "not at all the case" and 10 represents "completely the case". As shown in Figure 12, the Intrinsic Load (IL), representing task complexity and the learner's prior knowledge, averaged at 5.08 ( $SD = 0.42$ ). This mid-scale score indicates that participants found the task complexity to be moderately challenging, striking a balance between being too easy and overly difficult. The relatively low standard deviation suggests consistency in this perception across participants. Extraneous Load (EL), reflecting unnecessary cognitive burden imposed by instructional features, scored a mean of 2.75 ( $SD = 0.66$ ). This significantly lower score implies that Tutorly's instructional elements are well-designed, avoiding unnecessary complexity and cognitive overhead. The slightly higher standard deviation here might indicate some variation in how different users perceived the clarity of instructions or explanations. Germane Load (GL), associated with cognitive processes beneficial for learning, stood out with a mean of 8.05 ( $SD = 0.37$ ). This high score strongly suggests that Tutorly's instructional features effectively enhanced participants' learning and understanding. The low standard deviation indicates a strong consensus among users regarding the system's ability to promote meaningful cognitive processing. The combination of moderate IL, low EL, and high GL indicates that Tutorly could effectively adapt the task difficulty to students' mastery of skills, indicating that the student model and *CogApp* framework are well combined.

### 6.3 Tutorly's Failure Modes

**6.3.1 Error handling and rollback capability.** The usability study also revealed areas for potential enhancement, particularly in error handling and system robustness. Only 56.25% of participants agreed or strongly agreed that they could recover easily and quickly when they made a mistake during practice (Q14). Similarly, 62.5% found the system's error messages clearly indicated how to fix problems (Q13). P13 remarked: *Sometimes when I make a mistake, I don't want to see the correct answer directly. Undo and redo will be very helpful.*" This feedback suggests that improving error feedback and

1145 providing a “return to the previous step” could significantly enhance the learning experience. Additionally, integrating  
1146 more robust debugging tools and recovery mechanisms could further differentiate Tutorly from traditional video  
1147 learning and ChatGPT interactions. As P4 suggested: *It would be great if Tutorly could offer more detailed explanations*  
1148 *when my code doesn’t work, maybe even give more examples to practice.* Implementing these improvements could further  
1149 solidify Tutorly’s position as a comprehensive and user-friendly programming learning tool.  
1150

1151  
1152 6.3.2 *Viewing experience of video segments.* Sometimes, the video segments corresponding to specific learning goals  
1153 within the full video are too short or too long, affecting the learning experience. We noticed that because video creators  
1154 solved problems of varying difficulty at different speeds, a video clip can be as short as 12 seconds or as long as 348  
1155 seconds. One participant (P15) noted that it would be better if she could watch the video more coherently: *“Just from*  
1156 *my point of view, the Tutorials sometimes were cut a little suddenly. So, for people who don’t know data analysis so much,*  
1157 *it’s a challenge for them to give a reaction immediately.”* Similarly, one participant (P4) expressed that it is a waste of  
1158 time to watch the video author correct his own errors (common in unedited screencasts). Merging adjacent short video  
1159 clips or removing unnecessary parts from long video clips is a venue for future work.  
1160  
1161

## 1162 7 DISCUSSION

1163 In Tutorly, we employ the *CogApp* framework to realize a conversational LLM tutor system to assist in learning  
1164 from programming videos. We implement a novel pipeline and DSL to generate high-quality conversations while  
1165 allowing experts to specific key pedagogical strategies. Our user study findings highlight the benefits, limitations,  
1166 and opportunities of using Tutorly. Here, we elaborate on these limitations and discuss future opportunities for  
1167 improvements.  
1168

### 1169 7.1 Broader Utility

1170 Beyond a mentorship learning context, a compelling application of Tutorly is in pair programming. Pair programming is  
1171 a technique in which two individuals share a single computer as they work together to develop software. In traditional  
1172 pair programming, one person actively writes code (the “driver”), and the other person provides guidance, feedback, and  
1173 suggestions (the “navigator”). The capacity that Tutorly can adjust its role between driver and navigator could potentially  
1174 support some self-regulated learning phases—such as planning (setting goals), monitoring (tracking comprehension),  
1175 and reflection (evaluating progress). This could also position Tutorly as a tool that fosters self-directed learning (SDL) [7].  
1176 While SDL typically emphasizes independent goal-setting and resource management, Tutorly’s adaptability can help  
1177 learners formulate learning goals and bridge these practices with the structured support learners need to develop  
1178 autonomy. Like CoPilot, which provides code writing collaboration, we imagine a future version of Tutorly could  
1179 support collaborative brainstorming and problem-solving in open-ended domains.  
1180

1181 Second, Tutorly has strong potential in formal classroom settings. Since the pandemic, many courses have been  
1182 offered in a hybrid format with recorded lectures for students to watch asynchronously. Instructors and teaching  
1183 assistants could leverage Tutorly’s controllability to integrate learning tasks into video-based lessons, personalize  
1184 guidance for low-achieving students, and reduce their workload by supporting students in identifying, analyzing, and  
1185 solving problems. Further, teaching assistants might be able to personalize the prompt parameters in Tutorly to provide  
1186 better guidance and practice on hard-to-grasp concepts and for low-achieving students. Rather than accompanying  
1187 students to practice their ability to write code, Tutorly can also offset the burden on teaching assistants by helping  
1188 students to find problems, think about problems, and solve problems while watching videos.  
1189  
1190

## 1197 7.2 Limitations and Future Work

1198

1199

1200

1201 7.2.1 *Learning Gains Evaluation.* While pre/post-tests were conducted during the lab study, their results are not reported  
 1202 due to the study's time constraints and design limitations. Participants used Tutorly for less than an hour, which provided  
 1203 insufficient exposure to measure meaningful learning gains. Additionally, as the study employed a within-subject  
 1204 design, attributing differences between pre- and post-tests solely to Tutorly is challenging. We acknowledge that a more  
 1205 comprehensive evaluation of Tutorly's impact on learning requires a longer-term, between-subject study. A future  
 1206 large-scale study, ideally spanning a semester or more, will allow for a more comprehensive investigation into Tutorly's  
 1207 effects on learning outcomes.

1208

1209

1210

1211 7.2.2 *Adding Novel Interventions.* In Tutorly's conversational interactions, we allow experts to specify well-known  
 1212 intervention strategies such as multiple-choice-questions (MCQs) or fill in the blank. There is an opportunity to  
 1213 support more expressive interactions, such as allowing students to directly annotate over visualizations to engage in  
 1214 diagram construction, etc. However, our current implementation favors generalizability over integrating topic-specific  
 1215 techniques. Future research can investigate ways for LLMs to transform users' natural language description into a  
 1216 domain-specific language for domain-specific interaction formats. Alternately, the DSL can be extended to include  
 1217 techniques for generating interactive widgets that are interpretable for the student model.

1218

1219

1220

1221 7.2.3 *Cross-Platforms and Cross-Language Learning.* We deploy Tutorly as an extension in JupyterLab that provides a  
 1222 convenient platform for interactive data science and scientific computing. However, users may have preferences for  
 1223 an integrated development environment (IDE), such as using R Studio for R, PyCharm for Python, XCode for C++, or  
 1224 Visual Studio Code. Therefore, future work can extend Tutorly to multiple IDEs as plug-ins. A different scenario is  
 1225 when students are learning game development; they not only need IDE to write C# code but also need Unity to watch  
 1226 and manipulate. This will require cross-platform integration. More importantly, we imagine future iterations of Tutorly  
 1227 can take an input video in one language, such as R, and generate code and instructions in a different language, such as  
 1228 Python while keeping the underlying learning content the same.

1229

1230

1231

1232 7.2.4 *Supporting Applications in Diverse Domains and Video Structure.* Although our evaluation study provides valuable  
 1233 insights into Tutorly's usage in exploratory data analysis videos, this topic does not encompass all the programming  
 1234 videos. Participants also expressed interest in learning other topics, such as machine learning and game development.  
 1235 Our technical evaluation indicates that the current DSL structure can support conversation for these topics due to their  
 1236 reliance on text-based resources (e.g., code and slides) and the availability of comprehensive tutorials on platforms  
 1237 like YouTube, which provide sufficient training data for LLMs. Because Tutorly relies on text and code like in typical  
 1238 programming tutorials, lessons involving coding perform better than those that require learners to operate interactive  
 1239 software (e.g., block programming). Future enhancements to the framework can include Tutorly's ability to track UI for  
 1240 better generalizability.

1241

1242

1243

1244

1245

1246

1247

1248

On less common programming topics, such as financial model programming or bioinformatics algorithms, LLMs  
 may replicate errors from their training data and generate hallucinated statements not grounded in that data [9]. To  
 enhance Tutorly's versatility, future work should continue to explore Retrieval Augmented Generation (RAG) based  
 techniques [38]. This approach could enable Tutorly to access a broader, more specialized knowledge base, allowing it  
 to adapt more effectively to various programming domains without the need for extensive retraining.

1249 However, Tutorly's generalizability also depends on the structure and quality of the instructional videos it processes.  
1250 The system currently assumes a modular and relatively linear structure, where each segment corresponds to a discrete  
1251 learning goal. For videos that do not follow this structure – for instance, when creators frequently reference other  
1252 sections or provide overlapping content—the segmentation process might disrupt the instructional flow, potentially  
1253 losing critical context. Additionally, segment lengths and granularity can vary widely across unseen tutorials, which may  
1254 affect Tutorly's ability to generate meaningful prompts or interactions. To address these challenges, future iterations  
1255 of Tutorly could incorporate adaptive segmentation techniques that use temporal coherence, multimodal cues (e.g.,  
1256 audio and visual data), and user feedback to refine the alignment of segments with instructional goals. We also imagine  
1257 content creators can leverage Tutorly to provide more engaging viewing and learning experiences within their own  
1258 platforms to scale the utility of their content.  
1259

1260 Beyond programming, another important area for exploration is how Tutorly can support learning from everyday  
1261 instructional videos, such as those focused on cooking, makeup, or fitness. While there are existing works that assist  
1262 in learning from life-teaching videos [72], providing students with interactive feedback, rather than passive learning,  
1263 remains a challenge. Multi-modal LLMs offer a potential solution, enabling systems to monitor and analyze student  
1264 performance, facilitating a more dynamic and personalized learning experience in the future.  
1265

## 1266 8 CONCLUSION

1267 We developed Tutorly, a system for assisting students learn programming while watching online tutorial videos. While  
1268 using Tutorly, learners interact with an LLM-powered tutor through rich conversational interactions and follow the  
1269 guidance and feedback to meet their learning goals. Tutorly design is informed by the *CogApp* framework in learning  
1270 sciences, and backed by a student model to generate targeted pedagogical strategies to support effective learning. In  
1271 other words, Tutorly dynamically adapts its teaching strategies based on the observed learning progress of students.  
1272 Our prompting pipeline is generalizable across different topics, actions, and learning goals. Our user evaluation reveals  
1273 that Tutorly augments video-based learning experiences, is aligned with learning goals, and does not increase the  
1274 learner's cognitive workload. Using Tutorly, learners can effectively engage in self-directed learning to acquire and  
1275 apply the concepts and techniques contained in programming videos.  
1276

## 1277 REFERENCES

- 1278 [1] Shamim Akhter, Muhammad Kashan Javed, Syed Qasim Shah, and A Javaid. 2021. Highlighting the advantages and disadvantages of E-learning.  
1279 *Psychol. Educ* 58, 5 (2021), 1607–14.
- 1280 [2] Carlos Alario-Hoyos, C Delgado Kloos, Iria Estévez-Ayres, Carmen Fernández-Panadero, Jorge Blasco, Sergio Pastrana, and J Villena-Román. 2016.  
1281 Interactive activities: the key to learning programming with MOOCs. *Proceedings of the European Stakeholder Summit on Experiences and Best  
1282 Practices in and Around MOOCs, EMOOCs* 319 (2016).
- 1283 [3] Lorin W Anderson and David R Krathwohl. 2001. *A taxonomy for learning, teaching, and assessing: A revision of Bloom's taxonomy of educational  
1284 objectives: complete edition*. Addison Wesley Longman, Inc.
- 1285 [4] Pablo Arnau-González, Miguel Arevalillo-Herráez, Romina Albornoz-De Luise, and David Arnau. 2023. A methodological approach to enable natural  
1286 language interaction in an Intelligent Tutoring System. *Computer Speech Language* 81 (2023), 101516. <https://doi.org/10.1016/j.csl.2023.101516>
- 1287 [5] Anirudhan Badrinath, Frederic Wang, and Zachary Pardos. 2021. pyBKT: An Accessible Python Library of Bayesian Knowledge Tracing Models.  
1288 arXiv:2105.00385 [cs.MS] <https://arxiv.org/abs/2105.00385>
- 1289 [6] Brigid JS Barron, Daniel L Schwartz, Nancy J Vye, Allison Moore, Anthony Petrosino, Linda Zech, and John D Bransford. 2014. Doing with  
1290 understanding: Lessons from research on problem-and project-based learning. In *Learning through problem solving*. Psychology Press, 271–311.
- 1291 [7] Stephen D Brookfield. 2009. Self-directed learning. In *International handbook of education for the changing world of work: Bridging academic and  
1292 vocational learning*. Springer, 2615–2627.
- 1293 [8] John Seely Brown, Allan M. Collins, and Paul Duguid. 1989. Situated Cognition and the Culture of Learning. *Educational Researcher* 18 (1989), 32 –  
1294 42. <https://doi.org/10.3102/0013189X018001032>

- [9] Mikhail Burtsev, Martin Reeves, and Adam Job. 2023. The Working Limitations of Large Language Models. *MIT Sloan Management Review* 65, 1 (2023), 1–5.
- [10] Yining Cao, Hariharan Subramonyam, and Eytan Adar. 2022. VideoSticker: A Tool for Active Viewing and Visual Note-taking from Videos. In *Proceedings of the 27th International Conference on Intelligent User Interfaces* (Helsinki, Finland) (IUI '22). Association for Computing Machinery, New York, NY, USA, 672–690. <https://doi.org/10.1145/3490099.3511132>
- [11] Jaime R. Carbonell. 1970. AI in CAI: An Artificial-Intelligence Approach to Computer-Assisted Instruction. *IEEE Transactions on Man-Machine Systems* 11, 4 (1970), 190–202. <https://doi.org/10.1109/TMMS.1970.299942>
- [12] Victor Cassone. 2022. How to Use Deliberate Practice to Learn Programming More Efficiently. <https://www.freecodecamp.org/news/how-to-use-deliberate-practice-to-learn-programming-fast/>
- [13] Mehmet Celepkolu and Kristy Elizabeth Boyer. 2018. The Importance of Producing Shared Code Through Pair Programming. In *Proceedings of the 49th ACM Technical Symposium on Computer Science Education* (Baltimore, Maryland, USA) (SIGCSE '18). Association for Computing Machinery, New York, NY, USA, 765–770. <https://doi.org/10.1145/3159450.3159506>
- [14] ChatScope Development Team. 2024. ChatScope UI Kit. <https://chatscope.io> Accessed: 2024-04-03.
- [15] Debnath Chatterjee and Janet Corral. 2017. How to write well-defined learning objectives. *The journal of education in perioperative medicine: JEPM* 19, 4 (2017).
- [16] Yan Chen, Walter S. Lasecki, and Tao Dong. 2021. Towards Supporting Programming Education at Scale via Live Streaming. *Proc. ACM Hum.-Comput. Interact.* 4, CSCW3, Article 259 (Jan. 2021), 19 pages. <https://doi.org/10.1145/3434168>
- [17] Alan Y. Cheng, Meng Guo, Melissa Ran, Arpit Ranasaria, Arjun Sharma, Anthony Xie, Khuyen N. Le, Bala Vinaithirthan, Shihe (Tracy) Luan, David Thomas Henry Wright, Andrea Cuadra, Roy Pea, and James A. Landay. 2024. Scientific and Fantastical: Creating Immersive, Culturally Relevant Learning Experiences with Augmented Reality and Large Language Models. In *Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems* (Honolulu, HI, USA) (CHI '24). Association for Computing Machinery, New York, NY, USA, Article 275, 23 pages. <https://doi.org/10.1145/3613904.3642041>
- [18] Seulgi Choi, Hyewon Lee, Yoonjoo Lee, and Juho Kim. 2024. VIVID: Human-AI Collaborative Authoring of Vicarious Dialogues from Lecture Videos. In *Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems* (Honolulu, HI, USA) (CHI '24). Association for Computing Machinery, New York, NY, USA, Article 277, 26 pages. <https://doi.org/10.1145/3613904.3642867>
- [19] Allan Collins, John Seely Brown, Ann Holum, et al. 1991. Cognitive apprenticeship: Making thinking visible. *American educator* 15, 3 (1991), 6–11.
- [20] Myroslava Dzikovska, Natalie Steinhauser, Elaine Farrow, Johanna Moore, and Gwendolyn Campbell. 2014. BEETLE II: Deep natural language understanding and automatic feedback generation for intelligent tutoring in basic electricity and electronics. *International Journal of Artificial Intelligence in Education* 24, 3 (Sept. 2014), 284–332. <https://doi.org/10.1007/s40593-014-0017-9>
- [21] Travis Faas, Lynn Dombrowski, Alyson Young, and Andrew D. Miller. 2018. Watch Me Code: Programming Mentorship Communities on Twitch.Tv. *Proc. ACM Hum.-Comput. Interact.* 2, CSCW, Article 50 (nov 2018), 18 pages. <https://doi.org/10.1145/3274319>
- [22] Robert M Gagne, Walter W Wager, Katharine C Golas, John M Keller, and James D Russell. 2005. Principles of instructional design.
- [23] Ankit Gandhi, Arijit Biswas, Kundan Shrivastava, Ranjeet Kumar, Sahil Loomba, and Om Deshmukh. 2016. Easy Navigation through Instructional Videos using Automatically Generated Table of Content. In *Companion Publication of the 21st International Conference on Intelligent User Interfaces* (Sonoma, California, USA) (IUI '16 Companion). Association for Computing Machinery, New York, NY, USA, 92–96. <https://doi.org/10.1145/2876456.2879472>
- [24] Christine Geith and Karen Vignare. 2008. Access to Education with Online Learning and Open Educational Resources: Can They Close the Gap?. *Journal of asynchronous learning networks* 12, 1 (2008), 105–126.
- [25] Michail N. Giannakos. 2013. Exploring the video-based learning research: A review of the literature. *British Journal of Educational Technology* 44, 6 (2013), E191–E195. <https://doi.org/10.1111/bjet.12070>
- [26] Ralf Herbrich, Tom Minka, and Thore Graepel. 2006. TrueSkill™: A Bayesian Skill Rating System. In *Advances in Neural Information Processing Systems*, B. Schölkopf, J. Platt, and T. Hoffman (Eds.), Vol. 19. MIT Press. [https://proceedings.neurips.cc/paper\\_files/paper/2006/file/f44ee263952e65b3610b8ba51229df9-Paper.pdf](https://proceedings.neurips.cc/paper_files/paper/2006/file/f44ee263952e65b3610b8ba51229df9-Paper.pdf)
- [27] Hyoungwook Jin, Seonghee Lee, Hyungyu Shin, and Juho Kim. 2024. Teach AI How to Code: Using Large Language Models as Teachable Agents for Programming Education. In *Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems* (Honolulu, HI, USA) (CHI '24). Association for Computing Machinery, New York, NY, USA, Article 652, 28 pages. <https://doi.org/10.1145/3613904.3642349>
- [28] Enkelejda Kasneci, Kathrin Sessler, Stefan Küchemann, Maria Bannert, Daryna Dementieva, Frank Fischer, Urs Gasser, Georg Groh, Stephan Günemann, Eyke Hüllermeier, Stephan Krusche, Gitta Kutyniok, Tilman Michaeli, Claudia Nerdel, Jürgen Pfeffer, Oleksandra Poquet, Michael Sailer, Albrecht Schmidt, Tina Seidel, Matthias Stadler, Jochen Weller, Jochen Kuhn, and Gjergji Kasneci. 2023. ChatGPT for good? On opportunities and challenges of large language models for education. *Learning and Individual Differences* 103 (2023), 102274. <https://doi.org/10.1016/j.lindif.2023.102274>
- [29] Majeed Kazemitaabar, Runlong Ye, Xiaoning Wang, Austin Zachary Henley, Paul Denny, Michelle Craig, and Tovi Grossman. 2024. CodeAid: Evaluating a Classroom Deployment of an LLM-based Programming Assistant that Balances Student and Educator Needs. In *Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems* (Honolulu, HI, USA) (CHI '24). Association for Computing Machinery, New York, NY, USA, Article 650, 20 pages. <https://doi.org/10.1145/3613904.3642773>
- [30] Ada S. Kim and Amy J. Ko. 2017. A Pedagogical Analysis of Online Coding Tutorials. In *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education* (Seattle, Washington, USA) (SIGCSE '17). Association for Computing Machinery, New York, NY, USA, 321–326.

- 1353 <https://doi.org/10.1145/3017680.3017728>
- 1354 [31] Amy J. Ko, Thomas D. LaToza, Stephen Hull, Ellen A. Ko, William Kwok, Jane Quichocho, Harshitha Akkaraju, and Rishin Pandit. 2019. Teaching  
1355 Explicit Programming Strategies to Adolescents. In *Proceedings of the 50th ACM Technical Symposium on Computer Science Education* (Minneapolis,  
1356 MN, USA) (SIGCSE '19). Association for Computing Machinery, New York, NY, USA, 469–475. <https://doi.org/10.1145/3287324.3287371>
- 1357 [32] Mon-Lin Monica Ko and Christina Krist. 2019. Opening up curricula to redistribute epistemic agency: A framework for supporting science teaching.  
1358 *Science Education* 103, 4 (2019), 979–1010. <https://doi.org/10.1002/sce.21511> arXiv:<https://onlinelibrary.wiley.com/doi/pdf/10.1002/sce.21511>
- 1359 [33] Dimitrios Kravvaris and Katia Lida Kermanidis. 2019. Automatic point of interest detection for open online educational video lectures. *Multimedia  
1360 Tools Appl.* 78, 2 (Jan. 2019), 2465–2479. <https://doi.org/10.1007/s11042-018-6372-z>
- 1361 [34] Abhishek Kumar, Kushal Srivastava, Kuldeep Yadav, and Om Deshmukh. 2017. Multi-faceted Index driven Navigation for Educational Videos  
1362 in Mobile Phones. In *Proceedings of the 22nd International Conference on Intelligent User Interfaces* (Limassol, Cyprus) (IUI '17). Association for  
1363 Computing Machinery, New York, NY, USA, 357–361. <https://doi.org/10.1145/3025171.3025221>
- 1364 [35] Richard Catrambone Lauren E. Margulieux and Mark Guzdial. 2016. Employing subgoals in computer programming education. *Computer Science  
1365 Education* 26, 1 (2016), 44–67. <https://doi.org/10.1080/08993408.2016.1144429> arXiv:<https://doi.org/10.1080/08993408.2016.1144429>
- 1366 [36] Jimmie Leppink, Fred Paas, Cees Van der Vleuten, Tamara Gog, and Jeroen J. G. Van Merriënboer. 2013. Development of an instrument for measuring  
1367 different types of cognitive load. *Behavior research methods* 45 (04 2013). <https://doi.org/10.3758/s13428-013-0334-1>
- 1368 [37] James R. Lewis. 1992. Psychometric Evaluation of the Post-Study System Usability Questionnaire: The PSSUQ. *Proceedings of the Human Factors  
1369 Society Annual Meeting* 36, 16 (1992), 1259–1260. <https://doi.org/10.1177/154193129203601617> arXiv:<https://doi.org/10.1177/154193129203601617>
- 1370 [38] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim  
1371 Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Proceedings of the  
1372 34th International Conference on Neural Information Processing Systems* (Vancouver, BC, Canada) (NIPS '20). Curran Associates Inc., Red Hook, NY,  
1373 USA, Article 793, 16 pages.
- 1374 [39] Marcia C Linn. 1995. Designing computer learning environments for engineering and computer science: The scaffolded knowledge integration  
1375 framework. *Journal of Science Education and technology* 4 (1995), 103–126.
- 1376 [40] Ching (Jean) Liu, Chi-Lan Yang, Joseph Jay Williams, and Hao-Chuan Wang. 2019. NoteStruct: Scaffolding Note-taking while Learning from Online  
1377 Videos. In *Extended Abstracts of the 2019 CHI Conference on Human Factors in Computing Systems* (Glasgow, Scotland Uk) (CHI EA '19). Association  
1378 for Computing Machinery, New York, NY, USA, 1–6. <https://doi.org/10.1145/3290607.3312878>
- 1379 [41] Rongxin Liu, Carter Zenke, Charlie Liu, Andrew Holmes, Patrick Thornton, and David J. Malan. 2024. Teaching CS50 with AI: Leveraging Generative  
1380 Artificial Intelligence in Computer Science Education. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 1*  
1381 (Portland, OR, USA) (SIGCSE 2024). Association for Computing Machinery, New York, NY, USA, 750–756. <https://doi.org/10.1145/3626252.3630938>
- 1382 [42] Zhengyuan Liu, Stella Xin Yin, Carolyn Lee, and Nancy F. Chen. 2024. Scaffolding Language Learning via Multi-modal Tutoring Systems with  
1383 Pedagogical Instructions . In *2024 IEEE Conference on Artificial Intelligence (CAI)*. IEEE Computer Society, Los Alamitos, CA, USA, 1258–1265.  
1384 <https://doi.org/10.1109/CAI59869.2024.000223>
- 1385 [43] Zhengyuan Liu, Stella Xin Yin, Geyu Lin, and Nancy F. Chen. 2024. Personality-aware Student Simulation for Conversational Intelligent Tutoring  
1386 Systems. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung  
1387 Chen (Eds.). Association for Computational Linguistics, Miami, Florida, USA, 626–642. <https://doi.org/10.18653/v1/2024.emnlp-main.37>
- 1388 [44] Jakub Macina, Nico Daheim, Sankalan Chowdhury, Tamay Sinha, Manu Kapur, Iryna Gurevych, and Mrinmaya Sachan. 2023. MathDial: A  
1389 Dialogue Tutoring Dataset with Rich Pedagogical Properties Grounded in Math Reasoning Problems. In *Findings of the Association for Computational  
1390 Linguistics: EMNLP 2023*, Houda Bouamor, Juan Pino, and Kalika Bali (Eds.). Association for Computational Linguistics, Singapore, 5602–5621.  
1391 <https://doi.org/10.18653/v1/2023.findings-emnlp.372>
- 1392 [45] Christopher J. MacLellan and Kenneth R. Koedinger. 2022. Domain-General Tutor Authoring with Apprentice Learner Models. *International Journal  
1393 of Artificial Intelligence in Education* 32, 1 (March 2022), 76–117. <https://doi.org/10.1007/s40593-020-00214-2>
- 1394 [46] Angela Provitera McGlynn. 2005. Teaching millennials, our newest cultural cohort. *Education digest* 71, 4 (2005), 12.
- 1395 [47] Janet Metcalfe, Judy Xu, Matti Vuorre, Robert Siegler, Dylan Wiliam, and Robert A Bjork. 2024. Learning from errors versus explicit instruction in  
1396 preparation for a test that counts. *British Journal of Educational Psychology* (2024).
- 1397 [48] K. Mukhtar, K. Javed, M. Arooj, and A. Sethi. 2020. Advantages, Limitations and Recommendations for online learning during COVID-19 pandemic  
1398 era. *Pak J Med Sci* 36, COVID19-S4 (May 2020), S27–S31. <https://doi.org/10.12669/pjms.36.COVID19-S4.2785>
- 1399 [49] Medhini Narasimhan, Arsha Nagrani, Chen Sun, Michael Rubinstein, Trevor Darrell, Anna Rohrbach, and Cordelia Schmid. 2022. TL;DW?  
1400 Summarizing Instructional Videos with Task Relevance and Cross-Modal Saliency. In *Computer Vision – ECCV 2022*, Shai Avidan, Gabriel Brostow,  
1401 Moustapha Cissé, Giovanni Maria Farinella, and Tal Hassner (Eds.). Springer Nature Switzerland, Cham, 540–557.
- 1402 [50] Inc. NetEase Youdao. 2023. BCEmbedding: Bilingual and Crosslingual Embedding for RAG. <https://github.com/netease-youdao/BCEmbedding>.
- 1403 [51] Hyacinth S Nwana. 1990. Intelligent tutoring systems: an overview. *Artificial Intelligence Review* 4, 4 (1990), 251–277.
- 1404 [52] Benjamin D. Nye, Dillon Mee, and Mark G. Core. 2023. Generative Large Language Models for Dialog-Based Tutoring: An Early Consideration of  
1405 Opportunities and Concerns. In *LLM@AIED*. 78–88. <https://ceur-ws.org/Vol-3487/paper4.pdf>
- 1406 [53] Sharmaine Gek Teng Ong and Gwendoline Choon Lang Quek. 2023. Enhancing teacher-student interactions and student online engagement in an  
1407 online learning environment. *Learning Environments Research* 26, 3 (Oct. 2023), 681–707. <https://doi.org/10.1007/s10984-022-09447-5>
- 1408 [54] OpenAI. 2024. GPT-4 Technical Report. arXiv:2303.08774 [cs.CL] <https://arxiv.org/abs/2303.08774>

- [55] Stack Overflow. [n. d.]. Stack Overflow Developer Survey 2023. <https://survey.stackoverflow.co/2023/#section-learning-to-code-learning-how-to-code>
- [56] José Paladines and Jaime Ramirez. 2020. A Systematic Literature Review of Intelligent Tutoring Systems With Dialogue in Natural Language. *IEEE Access* 8 (2020), 164246–164267. <https://doi.org/10.1109/ACCESS.2020.3021383>
- [57] Roy D. Pea. 2004. The Social and Technological Dimensions of Scaffolding and Related Theoretical Concepts for Learning, Education, and Human Activity. *Journal of the Learning Sciences* 13, 3 (2004), 423–451. [https://doi.org/10.1207/s15327809jls1303\\_6](https://doi.org/10.1207/s15327809jls1303_6)
- [58] Oleksandra Poquet, Lisa Lim, Negin Mirriahi, and Shane Dawson. 2018. Video and learning: a systematic review (2007–2017). In *Proceedings of the 8th International Conference on Learning Analytics and Knowledge* (Sydney, New South Wales, Australia) (LAK '18). Association for Computing Machinery, New York, NY, USA, 151–160. <https://doi.org/10.1145/3170358.3170376>
- [59] Tine S Pöitz. 2010. Learning outcomes: What are they? Who defines them? When and where are they defined? *Educational assessment, evaluation and accountability* 22, 2 (2010), 119–137.
- [60] Cheng Ren, Zachary Pardos, and Zhi Li. 2024. Human-AI Collaboration Increases Skill Tagging Speed but Degrades Accuracy. arXiv:2403.02259 [cs.HC] <https://arxiv.org/abs/2403.02259>
- [61] Mitchel Resnick. 2024. Generative AI and Creative Learning: Concerns, Opportunities, and Choices. *An MIT Exploration of Generative AI* (mar 27 2024). <https://mit-genai.pubpub.org/pub/gj6eod3e>.
- [62] Peter Robe, Sandeep K. Kuttal, Jake AuBuchon, and Jacob Hart. 2022. Pair programming conversations with agents vs. developers: challenges and opportunities for SE community. In *Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering* (Singapore, Singapore) (ESEC/FSE 2022). Association for Computing Machinery, New York, NY, USA, 319–331. <https://doi.org/10.1145/3540250.3549127>
- [63] Marija Sablić, Ana Miroslavljević, and Alma Škugor. 2021. Video-Based Learning (VBL)—Past, Present and Future: an Overview of the Research Published from 2008 to 2019. *Technology, Knowledge and Learning* 26, 4 (Dec. 2021), 1061–1077. <https://doi.org/10.1007/s10758-020-09455-5>
- [64] Sami Sarsa, Paul Denny, Arto Hellas, and Juho Leinonen. 2022. Automatic Generation of Programming Exercises and Code Explanations Using Large Language Models. In *Proceedings of the 2022 ACM Conference on International Computing Education Research - Volume 1* (Lugano and Virtual Event, Switzerland) (ICER '22). Association for Computing Machinery, New York, NY, USA, 27–43. <https://doi.org/10.1145/3501385.3543957>
- [65] Omar Shaikh, Valentino Emil Chai, Michele Gelfand, Diyi Yang, and Michael S. Bernstein. 2024. Rehearsal: Simulating Conflict to Teach Conflict Resolution. In *Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems* (Honolulu, HI, USA) (CHI '24). Association for Computing Machinery, New York, NY, USA, Article 920, 20 pages. <https://doi.org/10.1145/3613904.3642159>
- [66] Zhecheng Sheng, Raymond Finzel, Michael Lucke, Sheena Dufresne, Maria Gini, and Serguei Pakhomov. 2023. A Dialogue System for Assessing Activities of Daily Living: Improving Consistency with Grounded Knowledge. In *Proceedings of the Third DialDoc Workshop on Document-grounded Dialogue and Conversational Question Answering*. Smaranda Muresan, Vivian Chen, Kennington Casey, Vandyke David, Dethlefs Nina, Inoue Koji, Ekstedt Erik, and Ultes Stefan (Eds.). Association for Computational Linguistics, Toronto, Canada, 68–79. <https://doi.org/10.18653/v1/2023.dialdoc-1.8>
- [67] Erin D. Solomon, Jana Hackathorn, Rachel E. Tennial, Kathryn Blankmeyer, and Amy M. Garczynski. 2011. Learning by Doing: An Empirical Study of Active Teaching Techniques. *The Journal of Effective Teaching* 11 (2011), 40–54. <https://doi.org/10.1037/e683152011-599> Institution: American Psychological Association.
- [68] Shashank Sonkar, Naiming Liu, Debshila Mallick, and Richard Baraniuk. 2023. CLASS: A Design Framework for Building Intelligent Tutoring Systems Based on Learning Science principles. In *Findings of the Association for Computational Linguistics: EMNLP 2023*. Houda Bouamor, Juan Pino, and Kalika Bali (Eds.). Association for Computational Linguistics, Singapore, 1941–1961. <https://doi.org/10.18653/v1/2023.findings-emnlp.130>
- [69] Makesh Narsimhan Sreedhar, Traian Rebedea, Shaona Ghosh, Jiaqi Zeng, and Christopher Parisien. 2024. CanTalkAboutThis: Aligning Language Models to Stay on Topic in Dialogues. In *Findings of the Association for Computational Linguistics: EMNLP 2024*. Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (Eds.). Association for Computational Linguistics, Miami, Florida, USA, 12232–12252. <https://doi.org/10.18653/v1/2024.findings-emnlp.713>
- [70] David Stroupe. 2014. Examining classroom science practice communities: How teachers and students negotiate epistemic agency and learn science-as-practice. *Science Education* 98, 3 (2014), 487–516. <https://doi.org/10.1002/sce.21112> Place: US Publisher: John Wiley & Sons.
- [71] John Sweller. 2021. Instructional design. In *Encyclopedia of evolutionary psychological science*. Springer, 4159–4163.
- [72] Anh Truong, Peggy Chi, David Salesin, Irfan Essa, and Maneesh Agrawala. 2021. Automatic Generation of Two-Level Hierarchical Tutorials from Instructional Makeup Videos. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems* (Yokohama, Japan) (CHI '21). Association for Computing Machinery, New York, NY, USA, Article 108, 16 pages. <https://doi.org/10.1145/3411764.3445721>
- [73] Ying-Jui Tseng, Yu-Hsin Lin, Gautam Yadav, Norman Bier, and Vincent Aleven. 2024. Curio: Enhancing STEM Online Video Learning Experience Through Integrated, Just-in-Time Help-Seeking. In *Technology Enhanced Learning for Inclusive and Equitable Quality Education: 19th European Conference on Technology Enhanced Learning, EC-TEL 2024, Krems, Austria, September 16–20, 2024, Proceedings, Part I* (Krems, Austria). Springer-Verlag, Berlin, Heidelberg, 437–451. [https://doi.org/10.1007/978-3-031-72315-5\\_30](https://doi.org/10.1007/978-3-031-72315-5_30)
- [74] John W Tukey et al. 1977. *Exploratory data analysis*. Vol. 2. Reading, MA.
- [75] Graham Upton and Ian Cook. 2014. *A dictionary of statistics 3e*. Oxford University Press, USA.
- [76] Josef Wachtler, Mohammad Khalil, Behnam Taraghi, and Martin Ebner. 2016. On using learning analytics to track the activity of interactive MOOC videos. In *LAK 2016 Workshop on Smart Environments and Analytics in Video-Based Learning*.

- 1457 [77] Ian Walk, Arnold Yim, Ed Novak, Charles Reiss, and Daniel Graham. 2020. Redesigning the Online Video Lecture Player to Promote Active Learning.  
1458 In *2020 IEEE Frontiers in Education Conference (FIE)*. 1–5. <https://doi.org/10.1109/FIE44824.2020.9273898>
- 1459 [78] Kuldeep Yadav, Ankit Gandhi, Arijit Biswas, Kundan Shrivastava, Saurabh Srivastava, and Om Deshmukh. 2016. ViZig: Anchor Points based  
1460 Non-Linear Navigation and Summarization in Educational Videos. In *Proceedings of the 21st International Conference on Intelligent User Interfaces*  
1461 (Sonoma, California, USA) (*IUI '16*). Association for Computing Machinery, New York, NY, USA, 407–418. <https://doi.org/10.1145/2856767.2856788>
- 1462 [79] Baoquan Zhao, Shujin Lin, Xiaonan Luo, Songhua Xu, and Ruomei Wang. 2017. A Novel System for Visual Navigation of Educational Videos Using  
1463 Multimodal Cues. In *Proceedings of the 25th ACM International Conference on Multimedia* (Mountain View, California, USA) (*MM '17*). Association  
1464 for Computing Machinery, New York, NY, USA, 1680–1688. <https://doi.org/10.1145/3123266.3123406>
- 1465 [80] Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, Yifan Du,  
1466 Chen Yang, Yushuo Chen, Zhipeng Chen, Jinhao Jiang, Ruiyang Ren, Yifan Li, Xinyu Tang, Zikang Liu, Peiyu Liu, Jian-Yun Nie, and Ji-Rong Wen.  
1467 2024. A Survey of Large Language Models. arXiv:2303.18223 [cs.CL] <https://arxiv.org/abs/2303.18223>
- 1468 [81] Serkan Şendağ, İlker Yakin, and Nuray Gedik. 2023. Fostering creative thinking skills through computer programming: Explicit or integrated  
1469 teaching? *Education and Information Technologies* 28 (02 2023), 1–20. <https://doi.org/10.1007/s10639-023-11629-4>
- 1470  
1471  
1472  
1473  
1474  
1475  
1476  
1477  
1478  
1479  
1480  
1481  
1482  
1483  
1484  
1485  
1486  
1487  
1488  
1489  
1490  
1491  
1492  
1493  
1494  
1495  
1496  
1497  
1498  
1499  
1500  
1501  
1502  
1503  
1504  
1505  
1506  
1507  
1508

1509 **A APPENDIX**

1510 **A.1 Participant Demographic Information**

1513	Participant ID	Occupation	Experience in EDA
1514	P1	Architect	Novice
1515	P2	Undergraduate Student	Novice
1516	P3	Hardware Engineer	Novice
1517	P4	Graduate Student	Novice
1518	P5	Graduate Student	Intermediate
1519	P6	Physician	Novice
1520	P7	Graduate Student	Novice
1521	P8	Product Manager	Novice
1522	P9	Banker	Novice
1523	P10	Software Engineer	Novice
1524	P11	Middle School Teacher	Novice
1525	P12	Hardware Engineer	Novice
1526	P13	Hardware Engineer	Novice
1527	P14	Graduate Student	Novice
1528	P15	Creative Artist	Novice
1529	P16	Graduate Student	Intermediate
1530			

1531 Table 4. Participant occupations and experience levels in EDA.

1535 **A.2 Algorithms in Tutorly**

1537 **A.2.1 Video Segmentation by Learning Goal.** Let  $V = \{v_1, v_2, \dots, v_T\}$  be the set of all sentences in the video transcript, where  $T$  is the total number of sentences, and  $L = \{l_1, l_2, \dots, l_M\}$  be the set of learning goals. The transcript is segmented into  $S = \{s_1, s_2, \dots, s_N\}$ , where  $s_n$  corresponds to one segment for one of the learning goals  $l_m$ .  $N > M$ , so that each learning goal has multiple segments. We define three functions executed sequentially in Algorithm 1, which mimics the cognitive steps an individual might take when classifying video clips semantically and provides a structured methodology for LLMs to segment video content by learning goals.

---

1545 **Algorithm 1** VSLG - Video Segmentation by Learning Goal

---

1546 **Require:** Video transcript sentences  $\mathcal{V}$ , Learning goals  $\mathcal{L}$

1547 **Ensure:** Video segments by learning goals  $\mathcal{S}$

- 1548 1:  $\mathcal{S}' = \{s'_1, s'_2, \dots, s'_N\} \leftarrow \text{Summarize}(\mathcal{V}, \mathcal{L})$   
 1549     ▷ summarizes transcript of each learning goal, yielding a set of summary points of all segments without  
 1550     timestamps.
- 1551 2:  $\mathcal{R} \leftarrow \text{Retrieve}(\mathcal{S}', \mathcal{V})$   
 1552     ▷ retrieves sentences from the video transcript for each summary point  $s'_n$ .
- 1553 3:  $\mathcal{S} \leftarrow \text{Rearrange}(\mathcal{R})$   
 1554     ▷ determines the range of segments based on the timestamps of sentences and sorts them by time.
-

1561 A.2.2 *Knowledge Summary and Pedagogical Move.* The pseudo program flow of knowledge summary and pedagogical  
 1562 move is as follows:  
 1563

---

**Algorithm 2** Knowledge Summary and Pedagogical Move
 

---

1564 **Require:** Video segments  $S$ , Cognitive Apprenticeship framework  $\mathcal{F}$ , code blocks  $C$ , student model for each knowledge  
 1565  $p_k$   
 1566 **Ensure:** Summarized knowledge  $K$  and teaching methods  $M$

1567 1: Initialize knowledge list  $K \leftarrow \emptyset$   
 1568 2: Initialize teaching method list  $M \leftarrow \emptyset$   
 1569 3: **function** SUMMARIZEKNOWLEDGE( $S$ )  
 1570 4:   **for** each segment  $s \in S$  **do**  
 1571     5:     $K_s \leftarrow$  Generate knowledge snippets for  $s$  with  $C_s$   
 1572     6:    Sort  $K_s$  in order of appearance within  $s$   
 1573     7:     $K \leftarrow K \cup K_s$   
 1574   8:   **end for**  
 1575   9:   **return**  $K$   
 1576 10: **end function**  
 1577 11: **function** PLANTEACHINGMETHODS( $K, C$ )  
 1578 12:   **for** each knowledge  $k \in K$  **do** ▷ Apply the principles  
 1579     13:   Determine the index, complexity, diversity of  $k$   
 1580     14:    $M_k \leftarrow \emptyset$  ▷ Initialize the set of methods for knowledge  $k$   
 1581     15:   **if**  $index \leq 1$  **then**  
 1582       16:     $M_k.add(Modeling/Scaffolding)$  ▷ Start with global skills and task overviews  
 1583     17:   **else if**  $p_k \leq 0.5$  **then**  
 1584       18:     $M_k.add(Scaffolding/Coaching/Articulation)$  ▷ Depends on which one has not been used  
 1585       19:     $M_k.add(Reflection)$  ▷ Encourage self-assessment  
 1586     20:   **else if**  $p_k > 0.5$  **then**  
 1587       21:     $M_k.remove(Scaffolding)$   
 1588       22:   **end if**  
 1589     23:    $M \leftarrow M \cup M_k$   
 1590   24:   **end for**  
 1591   25:   **return**  $M$   
 1592 26: **end function**  
 1593 27:  $K \leftarrow$  SUMMARIZEKNOWLEDGE( $S$ )  
 1594 28:  $M \leftarrow$  PLANTEACHINGMETHODS( $K, C$ )

---

1595  
 1596  
 1597  
 1598  
 1599  
 1600  
 1601  
 1602  
 1603  
 1604  
 1605  
 1606  
 1607  
 1608  
 1609  
 1610  
 1611  
 1612

1613    *DSL Structure for Chatbot Message Generation.* The pseudo program flow of generating the DSL structure for  
 1614    the chatbot message is as follows:  
 1615

---

**Algorithm 3** DSL Structure for Chatbot Message Generation
 

---

1616    **Require:** Methods Set  $\mathcal{M}$ , Actions  $\mathcal{A}$ , Interactions  $\mathcal{I}$ , Parameters list  $\mathcal{P}$ , student behavior  $\mathcal{B}$ , knowledge  $K$ , student  
 1617    model  $p_K$

1618    **Ensure:** Chatbot interaction

```

1619   1: function GETDSL( $a, i$ )
1620   2:    Description: Generates DSL components for  $\mathcal{M}$ .
1621   3:     $prompt \leftarrow$  Use  $i$  to generate the prompt for action  $a$ 
1622   4:     $needResponse \leftarrow$  Determine if requires user response
1623   5:     $\mathcal{P} \leftarrow$  Extract parameters required for the prompt
1624   6:    return { $prompt, needResponse, \mathcal{P}$ }
1625   7: end function
1626   8: Initialize message queue  $Q$ 
1627   9: for each  $m \in \mathcal{M}$  do
1628   10:    for each  $(a, i) \in \mathcal{A} \times \mathcal{I}$  do
1629   11:     { $prompt, needResponse, \mathcal{P}$ }  $\leftarrow$  GETDSL( $a, i$ )
1630   12:     Enqueue { $m, a, i, prompt, \mathcal{P}, needResponse$ } into  $Q$ 
1631   13:    end for
1632   14: end for
1633   15: while not  $Q.empty$  do
1634   16:     $message \leftarrow Q.dequeue$ 
1635   17:    Chatbot.send( $message.prompt, message.\mathcal{P}$ )
1636   18:    if  $message.needResponse$  is false then
1637   19:     continue
1638   20:    else
1639   21:     Chatbot.wait(Signalresponse( $t$ ))
1640   22:      $p_K \leftarrow$  UpdateStudentModel( $\mathcal{B}, K, p_K$ )
1641   23:    end if
1642   24: end while

```

---

1643

1644

1645

1646

1647

1648

1649

1650

1651

1652

1653

1654

1655

1656

1657

1658

1659

1660

1661

1662

1663

1664

▷ "Use [interaction] to + definition of the method"

1665 A.2.4 *Student Model Update.* Each skill contains four parameters, which are defined as follows:

- 1666 •  $p_{mastery}^{(t)}$ : The probability that the student has mastered the knowledge at time  $t$ .
- 1667 •  $p_{transit}$ : The probability of the student transitioning to a state of knowledge after an opportunity to apply it.
- 1668 •  $p_{slip}$ : The probability the student makes an error when applying known knowledge.
- 1669 •  $p_{guess}$ : The probability the student correctly applies unknown knowledge.

1670 Our system initialize the  $p_{mastery}^{(t)}$  to be aligned with the expert input. The other three parameters are all initialized  
 1671 to be 0.3.  $p_{mastery}^{(t)}$  is updated using Equation 1, based on the observed student behavior obs. During the practicing  
 1672 process, if students respond correctly, such as choosing a correct choice in the multiple-choice question or writing the  
 1673 correct code line, the observation is *Correct* and Tutorly updates the parameters of that correctly practiced knowledge.  
 1674

$$1675 p_{mastery}^{(t|obs)} = \begin{cases} \frac{p_{mastery}^{(t)} \times (1-p_{slip})}{p_{mastery}^{(t)} \times (1-p_{slip}) + (1-p_{mastery}^{(t)}) \times p_{guess}} & \text{obs = correct} \\ \frac{p_{mastery}^{(t)} \times p_{slip}}{p_{mastery}^{(t)} \times p_{slip} + (1-p_{mastery}^{(t)}) \times (1-p_{guess})} & \text{obs = incorrect} \end{cases} \quad (1)$$

1683 The pseudo program flow is as follows:

---

1684 **Algorithm 4** Student Model Update

---

1685 **Require:** student behavior  $\mathcal{B}$ , knowledge  $K$ , student model  $p_K$

1686 **Ensure:** the Student model gets updated

```

1687 1: function UPDATESTUDENTMODEL( $\mathcal{B}, K, p_K$ )
1688 2:   Description: Updates the student model.
1689 3:   Wait for Signal( $t$ ) before proceeding
1690 4:   for each knowledge  $k \in K$  do
1691 5:     if  $k$  is semantically similar to an element in  $K$  then
1692 6:        $p_k \leftarrow$  the parameter for  $k$  in  $p_K$ 
1693 7:       if  $\mathcal{B}$  is correct then
1694 8:         Update  $p_k$  based on correct behavior
1695 9:       else
1696 10:        Update  $p_k$  based on incorrect behavior
1697 11:       end if
1698 12:     else
1699 13:       Create new parameter  $p_k$  for  $k$ 
1700 14:       if  $\mathcal{B}$  is correct then
1701 15:         Initialize  $p_k$  based on correct behavior
1702 16:       else
1703 17:         Initialize  $p_k$  based on incorrect behavior
1704 18:       end if
1705 19:     end if
1706 20:   end for
1707 21:    $p_K \leftarrow p_K \cup p_k$ 
1708 22:   return updated  $p_K$ 
1709 23: end function
1710 24:  $p_K \leftarrow$  UPDATESTUDENTMODEL( $\mathcal{B}, K, p_K$ )

```

---

### 1717 A.3 Domain Specific Language Structure

1718 Below is an example of the DSL structure for the video segment “Visualize the data” when the student has a novice  
 1719 skills level.  
 1720

```

1721 [
1722   {
1723     "knowledge": "Comparing the median earnings of graduates across different
1724       major categories using a box plot to visualize the distribution and
1725       identify which categories have the highest and lowest earnings.",
1726     "actions": [
1727       {
1728         "method": "Modeling",
1729         "action": "Introduce the current task, encourage the student to focus
1730           on the video, and follow your steps using plain-text",
1731         "prompt": "[Use one sentence to tell the student the problem ({{
1732           knowledge}}) explored in the video][Use one sentence to encourage
1733           the student to follow your steps to solve this problem]",
1734         "interaction": "plain-text",
1735         "parameters": ["knowledge"]
1736       }
1737     ],
1738   },
1739   {
1740     "knowledge": "Use `fct_reorder` on `Major_category` to achieve an ordered
1741       factor level based on the `Median` because it helps in identifying the
1742       order of majors from lowest to highest earning.",
1743     "actions": [
1744       {
1745         "method": "Scaffolding",
1746         "action": "Demonstrate the current task and provide explanations of
1747           the concepts underlying the current step of the task using plain-
1748           text.",
1749         "prompt": "[Use one sentence to explain the {knowledge} to do,
1750           including what effect we want to achieve, why we do it, and what
1751           function we use to do it.]",
1752         "interaction": "plain-text",
1753         "parameters": ["knowledge"]
1754       },
1755       {
1756         "method": "Coaching",
1757         "action": "Use fill-in-blanks to guide the student through practice
1758           exercises, offering targeted hints and feedback.",
1759         "prompt": "[Use one sentence to prompt the student to fill in the {{
1760           code-line-with-blanks}} below to practice the {knowledge}][Provide
1761           a brief hint to help them through it]",
1762         "interaction": "fill-in-blanks",
1763         "parameters": ["code-line-with-blanks"]
1764       }
1765     ],
1766   },
1767   {
1768     "knowledge": "Use `geom_boxplot` on `Major_category` and `Median` to achieve
1769       a visual comparison of distributions across categories because it can
1770       reveal variations in median earnings among different major categories.",
1771   }
1772 }
```

```
1769     "actions": [
1770         {
1771             "method": "Scaffolding",
1772             "action": "Demonstrate the current task and provide explanations of
1773                 the concepts underlying the current step of the task using plain-
1774                 text.",
1775             "prompt": "[Use one sentence to explain the {knowledge} to do,
1776                 including what effect we want to achieve, why we do it, and what
1777                 function we use to do it.]",
1778             "interaction": "plain-text",
1779             "parameters": ["knowledge"]
1780         }
1781     ],
1782 },
1783 {
1784     "knowledge": "Use `coord_flip` to achieve a readable x-axis by flipping the
1785         coordinates because long category names can be difficult to read on a
1786         standard x-axis.",
1787     "actions": [
1788         {
1789             "method": "Scaffolding",
1790             "action": "Demonstrate the current task and provide explanations of
1791                 the concepts underlying the current step of the task using plain-
1792                 text.",
1793             "prompt": "[Use one sentence to explain the {knowledge} to do,
1794                 including what effect we want to achieve, why we do it, and what
1795                 function we use to do it.]",
1796             "interaction": "plain-text",
1797             "parameters": ["knowledge"]
1798         },
1799         {
1800             "method": "Reflection",
1801             "action": "Encourage students to review and debug their code using
1802                 show-code, and to reflect on the learning process by executing
1803                 the complete code block to verify their understanding.",
1804             "prompt": "[Use one sentence to let the student compare his answer
1805                 with the standard {code-block}][Use one sentence to encourage the
1806                 student to execute the complete code block to verify his
1807                 understanding]",
1808             "interaction": "show-code",
1809             "parameters": ["code-block"]
1810         }
1811     ]
1812 },
1813 },
1814 },
1815 },
1816 },
1817 },
1818 },
1819 },
1820 ]
```

**A.4 Action Set**

Table 5 shows the action set that generates prompts.

Move	Action	Prompt
<i>Programming-Related</i>		
Scaffolding	Demonstrate the current task and provide explanations of the concepts underlying the current step of the task using <b>plain-text</b> .	[Use one sentence to explain the {knowledge} at this step, such as what effect to achieve, why we do it, and what function we use to do it]
Coaching	Use <b>fill-in-blanks</b> to guide the student through practice exercises, offering targeted hints and feedback.	[Use one sentence to prompt the student to fill in the {code-line-with-blanks} below to practice the {knowledge}][Provide a brief hint to help them through it]
Articulation	Use <b>plain-text</b> to allow students to articulate their understanding of knowledge.	[Use one sentence to ask the student to explain their understanding and reasoning about {knowledge}, such as articulate why make this kind of visualization rather than others]
Reflection	Encourage students to review and debug their code using <b>show-code</b> and to reflect on the learning process by executing the complete code block to verify their understanding.	[Use one sentence to let the student compare their answer with the standard {code-block}][Use one sentence to encourage the student to execute the complete code block to verify their understanding]
<i>Concept-Related</i>		
Scaffolding	Provide structured guidance through <b>plain-text</b> as the student works on the task to learn the {knowledge}.	[Use no more than three sentences to guide the student step by step on how to learn and apply the {knowledge}, the student has made the visualization]
Coaching	Use <b>multiple-choice</b> to observe the student's approach to tasks, offering feedback to guide learning.	Propose a multiple-choice question for the student to understand the {knowledge}, such as what could be the potential reason behind the pattern
Articulation	Encourage students to use <b>interaction</b> to verbally explain their thought process and reasoning behind their observations and conclusions.	[Use one sentence to ask the student to explain their understanding and reasoning about {knowledge}, such as articulate what patterns they found in the chart]
Reflection	Encourage students to use <b>plain-text</b> to self-evaluate their performance, identifying strengths and areas for improvement.	[Use one sentence to give feedback on the {student-answer}][Use one sentence to tell the student if any additional steps could confirm their choice][Ask the student to remember the choice and see if it makes sense as they watch the rest of the video]

Table 5. Programming and concept-related cognitive apprentice moves in teaching programming and concepts. The bold terms are interactions used in prompts. Parameters are quoted in curly brackets.

**A.5 Prompts for LLMs**

Below are the prompts we fed into the ChatGPT API to perform each of Tutorly's key algorithms.

Table 6. Prompt templates used in Video Segmentation, Knowledge Summarization and Teaching Methods Arrangement. The few-shot examples provides output format limitations.

Tasks	Prompt Template	Few-shot examples
Video Segmentation: Get Summary	Here is a video transcript about {video_topic}. Summarize the video content that corresponds to each given learning goal. The transcript is not necessarily arranged in the order in which the learning goals are defined and can contain multiple segments with the same learning goal. The script may contain only some of the learning goals. Do not include a summary of learning goals that do not exist in the transcript. Increase the granularity. For example, if the video author creates two different visualizations, they should be summarized into two points.	[ ("Introduction", summary), ("Load data/packages", summary), ("Understand the dataset", summary), ("Visualize the data", summary), ("Interpret the chart", summary), ("Visualize the data", summary), ("Interpret the chart", summary), ("Preprocess the data", summary), ...] ]
Knowledge Summarize	The following {video_type} video transcript is about a learning goal: {learning_goal}. Summarize the declarative and procedural knowledge in the video transcript. Each piece of knowledge should follow this format: Procedural knowledge: "To achieve/understand + [specific goal/outcome] + one need to + [general actions/processes] + [additional details] + and consider/use + [relevant factors/tools]." The [general actions/processes] should be quoted in a pair of & sign. Declarative knowledge: "[Subject] + [verb phrase] + that + [independent clause]."	[ "Procedural knowledge: To understand the distribution of earnings by college major, one need to \&examine the histogram and identify overall trend or extreme values\&, and consider whether high earnings are due to the field's financial reward or influenced by low sample size or high variation." ...] ]
Teaching Methods Arrangement	You are an expert mentor who is good at arranging teaching methods to help students learn from a video about {video_type}. You are teaching students to learn knowledge for {learning_obj} using the Cognitive Apprenticeship framework. Your task is to choose proper Cognitive Apprenticeship methods to teach the student the given knowledge. The input knowledge list contains the declarative and procedural knowledge in the video. The input student mastery level list has a one-to-one correspondence with the knowledge, representing the student's mastery of each knowledge. You need to use the global skills first, then the local skills principle to rearrange the order of knowledge to learn.	[ {{ "knowledge": "Declarative knowledge: The median income by college major shows that majors earn a median income of over \\$30K right out of college.", "method": ["Coaching"] }}, ...] ]

1925 1926 1927 1928 1929	<b>Tasks</b>	<b>Zero-shot Prompt Template</b>
1930 1931 1932 1933 1934 1935 1936 1937 1938 1939 1940 1941 1942 1943 1944	Video Segmentation: Get Sentences	Here is a transcript of the video about exploratory data analysis and a summary of one paragraph of the video transcript. Find the continuous sentence in the given transcript that corresponds to the summary. Don't add punctuation or capitalization that are not in the original transcript.
1945 1946 1947 1948 1949 1950 1951 1952 1953 1954 1955 1956	Video Segmentation: Get Timestamp	Here is a video transcript about exploratory data analysis and a continuous part of it. Find out the timestamp range corresponding to this part.
1957 1958 1959 1960 1961 1962 1963 1964 1965 1966 1967 1968 1969 1970 1971 1972 1973 1974 1975 1976	Conversation Simulation in the Ablation Study	<p>Your task is to simulate the conversation between a teacher and a student. Assume you are a teacher and also a student at the same time.</p> <p>As a teacher, you will use the information in the video to guide the student, such as the knowledge, video transcript, and code blocks.</p> <p>As a student, you will follow the instructions of the teacher to respond and sometimes ask questions about things that do not understand.</p> <p>Teacher's instruction should cover all knowledge while also needs to answer the student's questions.</p> <p>When the student responds to the teacher's question or tasks, the student has the probability to make mistakes. The teacher should give feedback to the student's response and guide the student to the correct answer.</p>
1977 1978 1979 1980 1981 1982 1983 1984 1985 1986 1987 1988 1989 1990 1991 1992 1993 1994 1995 1996 1997 1998 1999 2000 2001 2002 2003 2004 2005 2006 2007 2008 2009 2010 2011 2012 2013 2014 2015 2016 2017 2018 2019 2020 2021 2022 2023 2024 2025 2026 2027 2028 2029 2030 2031 2032 2033 2034 2035 2036 2037 2038 2039 2040 2041 2042 2043 2044 2045 2046 2047 2048 2049 2050 2051 2052 2053 2054 2055 2056 2057 2058 2059 2060 2061 2062 2063 2064 2065 2066 2067 2068 2069 2070 2071 2072 2073 2074 2075 2076 2077 2078 2079 2080 2081 2082 2083 2084 2085 2086 2087 2088 2089 2090 2091 2092 2093 2094 2095 2096 2097 2098 2099 20100 20101 20102 20103 20104 20105 20106 20107 20108 20109 20110 20111 20112 20113 20114 20115 20116 20117 20118 20119 20120 20121 20122 20123 20124 20125 20126 20127 20128 20129 20130 20131 20132 20133 20134 20135 20136 20137 20138 20139 20140 20141 20142 20143 20144 20145 20146 20147 20148 20149 20150 20151 20152 20153 20154 20155 20156 20157 20158 20159 20160 20161 20162 20163 20164 20165 20166 20167 20168 20169 20170 20171 20172 20173 20174 20175 20176 20177 20178 20179 20180 20181 20182 20183 20184 20185 20186 20187 20188 20189 20190 20191 20192 20193 20194 20195 20196 20197 20198 20199 20200 20201 20202 20203 20204 20205 20206 20207 20208 20209 20210 20211 20212 20213 20214 20215 20216 20217 20218 20219 20220 20221 20222 20223 20224 20225 20226 20227 20228 20229 20230 20231 20232 20233 20234 20235 20236 20237 20238 20239 20240 20241 20242 20243 20244 20245 20246 20247 20248 20249 20250 20251 20252 20253 20254 20255 20256 20257 20258 20259 20260 20261 20262 20263 20264 20265 20266 20267 20268 20269 20270 20271 20272 20273 20274 20275 20276 20277 20278 20279 20280 20281 20282 20283 20284 20285 20286 20287 20288 20289 20290 20291 20292 20293 20294 20295 20296 20297 20298 20299 202000 202001 202002 202003 202004 202005 202006 202007 202008 202009 202010 202011 202012 202013 202014 202015 202016 202017 202018 202019 202020 202021 202022 202023 202024 202025 202026 202027 202028 202029 202030 202031 202032 202033 202034 202035 202036 202037 202038 202039 202040 202041 202042 202043 202044 202045 202046 202047 202048 202049 202050 202051 202052 202053 202054 202055 202056 202057 202058 202059 202060 202061 202062 202063 202064 202065 202066 202067 202068 202069 202070 202071 202072 202073 202074 202075 202076 202077 202078 202079 202080 202081 202082 202083 202084 202085 202086 202087 202088 202089 202090 202091 202092 202093 202094 202095 202096 202097 202098 202099 2020100 2020101 2020102 2020103 2020104 2020105 2020106 2020107 2020108 2020109 2020110 2020111 2020112 2020113 2020114 2020115 2020116 2020117 2020118 2020119 2020120 2020121 2020122 2020123 2020124 2020125 2020126 2020127 2020128 2020129 2020130 2020131 2020132 2020133 2020134 2020135 2020136 2020137 2020138 2020139 2020140 2020141 2020142 2020143 2020144 2020145 2020146 2020147 2020148 2020149 2020150 2020151 2020152 2020153 2020154 2020155 2020156 2020157 2020158 2020159 2020160 2020161 2020162 2020163 2020164 2020165 2020166 2020167 2020168 2020169 2020170 2020171 2020172 2020173 2020174 2020175 2020176 2020177 2020178 2020179 2020180 2020181 2020182 2020183 2020184 2020185 2020186 2020187 2020188 2020189 2020190 2020191 2020192 2020193 2020194 2020195 2020196 2020197 2020198 2020199 2020200 2020201 2020202 2020203 2020204 2020205 2020206 2020207 2020208 2020209 2020210 2020211 2020212 2020213 2020214 2020215 2020216 2020217 2020218 2020219 2020220 2020221 2020222 2020223 2020224 2020225 2020226 2020227 2020228 2020229 2020230 2020231 2020232 2020233 2020234 2020235 2020236 2020237 2020238 2020239 2020240 2020241 2020242 2020243 2020244 2020245 2020246 2020247 2020248 2020249 2020250 2020251 2020252 2020253 2020254 2020255 2020256 2020257 2020258 2020259 2020260 2020261 2020262 2020263 2020264 2020265 2020266 2020267 2020268 2020269 2020270 2020271 2020272 2020273 2020274 2020275 2020276 2020277 2020278 2020279 2020280 2020281 2020282 2020283 2020284 2020285 2020286 2020287 2020288 2020289 2020290 2020291 2020292 2020293 2020294 2020295 2020296 2020297 2020298 2020299 2020300 2020301 2020302 2020303 2020304 2020305 2020306 2020307 2020308 2020309 2020310 2020311 2020312 2020313 2020314 2020315 2020316 2020317 2020318 2020319 2020320 2020321 2020322 2020323 2020324 2020325 2020326 2020327 2020328 2020329 2020330 2020331 2020332 2020333 2020334 2020335 2020336 2020337 2020338 2020339 2020340 2020341 2020342 2020343 2020344 2020345 2020346 2020347 2020348 2020349 2020350 2020351 2020352 2020353 2020354 2020355 2020356 2020357 2020358 2020359 2020360 2020361 2020362 2020363 2020364 2020365 2020366 2020367 2020368 2020369 2020370 2020371 2020372 2020373 2020374 2020375 2020376 2020377 2020378 2020379 2020380 2020381 2020382 2020383 2020384 2020385 2020386 2020387 2020388 2020389 2020390 2020391 2020392 2020393 2020394 2020395 2020396 2020397 2020398 2020399 2020400 2020401 2020402 2020403 2020404 2020405 2020406 2020407 2020408 2020409 2020410 2020411 2020412 2020413 2020414 2020415 2020416 2020417 2020418 2020419 2020420 2020421 2020422 2020423 2020424 2020425 2020426 2020427 2020428 2020429 2020430 2020431 2020432 2020433 2020434 2020435 2020436 2020437 2020438 2020439 2020440 2020441 2020442 2020443 2020444 2020445 2020446 2020447 2020448 2020449 2020450 2020451 2020452 2020453 2020454 2020455 2020456 2020457 2020458 2020459 2020460 2020461 2020462 2020463 2020464 2020465 2020466 2020467 2020468 2020469 2020470 2020471 2020472 2020473 2020474 2020475 2020476 2020477 2020478 2020479 2020480 2020481 2020482 2020483 2020484 2020485 2020486 2020487 2020488 2020489 2020490 2020491 2020492 2020493 2020494 2020495 2020496 2020497 2020498 2020499 2020500 2020501 2020502 2020503 2020504 2020505 2020506 2020507 2020508 2020509 2020510 2020511 2020512 2020513 2020514 2020515 2020516 2020517 2020518 2020519 2020520 2020521 2020522 2020523 2020524 2020525 2020526 2020527 2020528 2020529 2020530 2020531 2020532 2020533 2020534 2020535 2020536 2020537 2020538 2020539 2020540 2020541 2020542 2020543 2020544 2020545 2020546 2020547 2020548 2020549 2020550 2020551 2020552 2020553 2020554 2020555 2020556 2020557 2020558 2020559 2020560 2020561 2020562 2020563 2020564 2020565 2020566 2020567 2020568 2020569 2020570 2020571 2020572 2020573 2020574 2020575 2020576 2020577 2020578 2020579 2020580 2020581 2020582 2020583 2020584 2020585 2020586 2020587 2020588 2020589 2020590 2020591 2020592 2020593 2020594 2020595 2020596 2020597 2020598 2020599 2020600 2020601 2020602 2020603 2020604 2020605 2020606 2020607 2020608 2020609 2020610 2020611 2020612 2020613 2020614 2020615 2020616 2020617 2020618 2020619 2020620 2020621 2020622 2020623 2020624 2020625 2020626 2020627 2020628 2020629 2020630 2020631 2020632 2020633 2020634 2020635 2020636 2020637 2020638 2020639 2020640 2020641 2020642 2020643 2020644 2020645 2020646 2020647 2020648 2020649 2020650 2020651 2020652 2020653 2020654 2020655 2020656 2020657 2020658 2020659 2020660 2020661 2020662 2020663 2020664 2020665 2020666 2020667 2020668 2020669 2020670 2020671 2020672 2020673 2020674 2020675 2020676 2020677 2020678 2020679 2020680 2020681 2020682 2020683 2020684 2020685 2020686 2020687 2020688 2020689 2020690 2020691 2020692 2020693 2020694 2020695 2020696 2020697 2020698 2020699 2020700 2020701 2020702 2020703 2020704 2020705 2020706 2020707 2020708 2020709 2020710 2020711 2020712 2020713 2020714 2020715 2020716 2020717 2020718 2020719 2020720 2020721 2020722 2020723 2020724 2020725 2020726 2020727 2020728 2020729 2020730 2020731 2020732 2020733 2020734 2020735 2020736 2020737 2020738 2020739 2020740 2020741 2020742 2020743 2020744 2020745 2020746 2020747 2020748 2020749 2020750 2020751 2020752 2020753 2020754 2020755 2020756 2020757 2020758 2020759 2020760 2020761 2020762 2020763 2020764 2020765 2020766 2020767 2020768 2020769 2020770 2020771 2020772 2020773 2020774 2020775 2020776 2020777 2020778 2020779 2020780 2020781 2020782 2020783 2020784 2020785 2020786 2020787 2020788 2020789 2020790 2020791 2020792 2020793 2020794 2020795 2020796 2020797 2020798 2020799 2020800 2020801 2020802 2020803 2020804 2020805 2020806 2020807 2020808 2020809 2020810 2020811 2020812 2020813 2020814 2020815 2020816 2020817 2020818 2020819 2020820 2020821 2020822 2020823 2020824 2020825 2020826 2020827 2020828 2020829 2020830 2020831 2020832 2020833 2020834 2020835 2020836 2020837 2020838 2020839 2020840 2020841 2020842 2020843 2020844 2020845 2020846 2020847 2020848 2020849 2020850 2020851 2020852 2020853 2020854 2020855 2020856 2020857 2020858 2020859 2020860 2020861 2020862 2020863 2020864 2020865 2020866 2020867 2020868 2020869 2020870 2020871 2020872 2020873 2020874 2020875 2020876 2020877 2020878 2020879 2020880 2020881 2020882 2020883 2020884 2020885 2020886 2020887 2020888 2020889 2020890 2020891 2020892 2020893 2020894 2020895 2020896 2020897 2020898 2020899 2020900 2020901 2020902 2020903 2020904 2020905 2020906 2020907 2020908 2020909 2020910 2020911 2020912 2020913 2020914 2020915 2020916 2020917 2020918 2020919 2020920 2020921 2020922 2020923 202092		