Designing and Evaluating AI-Generated Multimodal Analogy-Based Explanations

Yuri Noviello(✉)

Delft University of Technology, Delft, The Netherlands
y.noviello@tudelft.nl

Abstract. Effectively teaching programming concepts remains a persistent challenge, as conventional approaches often struggle to make abstract ideas accessible and engaging. Analogies provide a powerful means to simplify these concepts, and when augmented with multimodal resources, such as video animations, can significantly enhance learner engagement and comprehension. This research leverages Large Language Models (LLMs) and a structured animation workflow to automate the generation of analogy-driven explanations and visualizations for programming topics. Preliminary findings indicate that while LLMs can produce creative and pedagogically relevant analogies, careful decomposition and validation steps are critical for ensuring clarity, correctness, and learning alignment. We will conduct controlled evaluations to compare the effectiveness of AI-generated analogy-driven materials against traditional instructional methods. We also explore domain-specific personalization, adapting analogies to the learner's familiar domains and background.

Keywords: Analogy Generation · Notional Machine · Animation Generation · Programming Education · Artificial Intelligence

1 Introduction

Traditional programming education methods often fail to engage learners and convey abstract ideas meaningfully. A proven strategy to mitigate these issues is to employ metaphors and analogies: by anchoring new and unfamiliar topics in learners' existing knowledge, analogies can render challenging material more intuitive and relatable [7]. When these analogies are augmented with multi-modal elements, such as video animations, the learning experience becomes potentially more dynamic, accommodating diverse learning styles and boosting both engagement and conceptual understanding [6]. Despite their pedagogical benefits, creating analogy-driven multimedia content remains time-intensive. Educators not only need deep expertise in programming, but also creative skills in animation and media production, resources that can be scarce or challenging to integrate into teaching schedules [11].

The rapid evolution of Artificial Intelligence (AI), especially in the area of Natural Language Processing (NLP), presents an opportunity to automate and

A. I. Cristea et al. (Eds.): AIED 2025, CCIS 2590, pp. 446–451, 2025.
https://doi.org/10.1007/978-3-031-99261-2_51

enhance educational content creation. AI has already proven effective in personalizing learning experiences, offering data-driven assessments, and providing on-demand tutoring in various STEM fields [4,8]. In programming education, AI-driven tools have demonstrated the ability to generate adaptive, context-sensitive feedback, but most systems still emphasize syntax correction or code completion rather than conceptual clarity delivered through analogies or metaphors.

There is no widely accessible solution that generates analogy-driven and multi-modal instructional materials to support educators. Such a solution would hold significant value for both the education and NLP communities by demonstrating how state-of-the-art language models methodologies can be applied to enhance pedagogy and learning outcomes. This research aims to develop an AI-based system that automates the generation of analogy-driven explanations and corresponding video animations for programming concepts, by leveraging the capabilities of Large Language Models and established educational design principles.

1.1 Research Questions

RQ1. How to design an AI tool that effectively generates analogies and animations for programming education, when evaluated against established analogy-quality metrics (e.g., *accuracy, accessibility, comprehensiveness*) and instructional design standards (e.g., *engagement, clarity, concept alignment*)?
RQ2. How do human analogy's evaluations (e.g., *student and teacher feedback, usability studies*) compare with automated evaluation strategies?
RQ3. How do AI-generated analogies and animations enhance students' comprehension and retention of complex programming concepts compared to traditional text-based or static visual materials, as measured through pre/post-tests and qualitative feedback?
RQ4. How does AI-driven personalized content generation, tailored to specific domains or learner profiles, affects student engagement, learning outcomes, and cognitive load, compared to generic, non-personalized instructional approaches?

2 Related Work

Analogies have long been recognized as powerful pedagogical tools for making abstract or complex ideas more accessible by linking them to familiar concepts. In programming education, these devices are frequently employed to simplify intricate topics. The value of analogies is further amplified when presented through multimodal channels, including visuals and animations. Animations can effectively illustrate dynamic processes and boost student engagement. The use of multimodal learning material has proven especially promising for fostering deeper comprehension and retention in educational contexts [3].

Metaphor and analogy processing represent a longstanding challenge in NLP. Because they rely on figurative rather than literal language, traditional systems have struggled to handle their semantic and contextual nuances [10].

More recently, the emergence of LLMs has opened up new avenues for tackling metaphor understanding and analogy generation at scale [2,13]. Empirical studies also point to the educational potential of such AI-generated analogies, underscoring their value for explaining programming concepts to learners [1].

Beyond the broad definitions of metaphor and analogy, notional machines offer another conceptual tool to aid understanding of programming languages and environments [5]. A notional machine is a pedagogical abstraction that emphasizes particular aspects of a language's semantics while omitting non-essential details. The goal is to reduce the cognitive load and help learners form accurate mental models of how programs execute. Since notional machine must align with the actual semantics of the targeted programming language to avoid introducing misconceptions, recent work formalizes soundness criteria enabling rigorous analysis of notional machines and the identification of inconsistencies [9].

3 Research Methods

This study employs a multi-phase methodology to develop, evaluate, and refine an AI-driven system that generates analogies and video animations for programming education. The research is structured into four main phases.

Phase 1: Tool Development and Preliminary Evaluation
In the initial phase, we build the core AI-based tool capable of producing analogy-driven textual explanations and accompanying video animations. The system leverages Large Language Models (LLMs) to generate analogies that simplify programming concepts and then transforms these explanations into short, explanatory animations. A small-scale, preliminary evaluation will follow, involving expert educators who will assess the output's accuracy, accessibility, and comprehensiveness, while students will provide feedback on engagement, clarity, and cognitive load.

Phase 2: Automated Evaluation Pipeline
Building on the insights from the preliminary evaluation, the next phase focuses on creating an AI-driven evaluation model designed to assess the quality of generated analogies and animations. The goal is to achieve expert-level reliability in judging whether the explanations are both pedagogically sound and semantically accurate. By automating quality checks, we aim to streamline iterative refinement and ensure consistently high-quality outputs.

Phase 3: Large-Scale Controlled Study
In this phase, the tool's effectiveness is rigorously tested through a controlled experiment with a larger group of students. We will compare the impact of AI-generated analogy and animations to that of traditional instructional materials. Evaluation metrics will include *comprehension* and *retention*, measured with pre and post-tests designed to calculate learning gains.

Phase 4: Domain Personalization and Extended Evaluation
In the final phase, we introduce personalization features that enable learners to select familiar domains (such as *sports*, *music*, or *gaming*) to contextualize analogies. By tailoring explanations to individual interests and background knowledge, we hypothesize that learners will experience a reduced cognitive load and increased engagement. This phase will include an extensive evaluation of how domain-specific analogies influence learning outcomes, comparing them against generic analogies in terms of knowledge retention and user satisfaction.

4 Current Work

Analogy Generation
Our ongoing efforts center on testing state-of-the-art LLMs to produce analogies for foundational programming topics Table 1. Our experiments show that, while LLMs can generate creative and intuitive analogies, they often require careful prompt engineering and iterative refinement to maximize clarity and correctness. We plan to systematically investigate various strategies and model configurations, with the goal of identifying best practices for generating analogies that are neither too domain-specific nor too general.

Table 1. LLM-generated analogy illustrating the topic *Recursion*

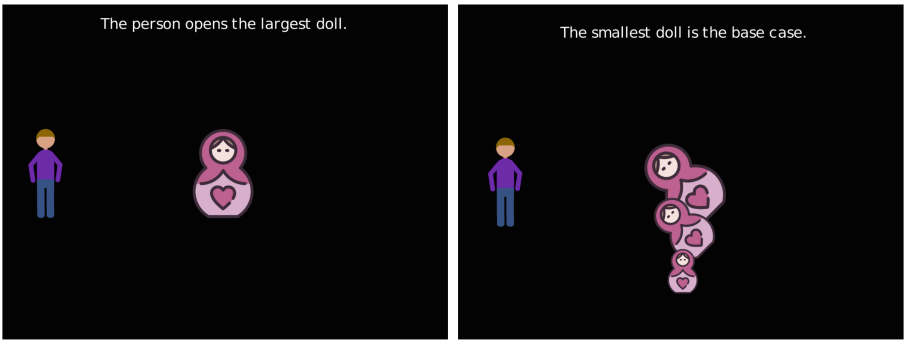
Topic:*Recursion*
Analogy: Imagine a set of Russian nesting dolls. You want to open the smallest doll inside, but to get there, you must first open each larger doll one by one. Each doll represents a step in the problem, and opening it is like calling the function. The smallest doll (the base case) is the point where you stop opening because there's nothing smaller inside. Once you reach the smallest doll, you start putting the dolls back together in reverse order (unwinding the recursion), until all the dolls are reassembled, and the problem is fully solved.

Animation Generation
In parallel, we have conducted several experiments using `manim` [12] as our animation framework Fig. 1. Directly prompting an LLM to generate a complete `manim` script frequently leads to syntactic and logical errors, reflecting the complexity of scene composition. To address these challenges, we devised a multi-stage pipeline:

1. Identify discrete visual components (e.g., shapes, text) tied to each part of the analogy.
2. Prompt the LLM to produce code snippets for individual components rather than the entire scene at once and validate these snippets to ensure syntactic correctness.

3. Merge validated code elements into a cohesive animation sequence, resolving any layout conflicts or timing issues through iterative checks.

Additionally, we developed a self-validation mechanism that detects semantic inconsistencies within the key frames of a video. Leveraging a Visual Large Language Model (vLLM), the system analyses these frames, automatically correcting content misalignment and improving the overall quality of the animation.



(a) (b)

Fig. 1. Frames extracted from LLM-generated animation explaining *Recursion* as a *Set of Russian Dolls*. 1a The function starts by opening the largest doll.1b The *recursion* reaches the `base case` when the smallest doll is found, and no more dolls remain to be opened.

Evaluation

Currently, we are conducting an initial round of evaluations by collecting expert and student annotations on a predefined set of textual and video-based analogies. Domain experts rate the output on established analogy evaluation metrics, such as *conceptual accuracy* and *soundness*. Students' feedback will be used to measure engagement, usability, and cognitive load. These assessments will help calibrate the automated evaluation model for the next phase of the research.

References

1. Bernstein, S., et al.: "Like a Nesting Doll": analyzing recursion analogies generated by CS students using large language models. In: Proceedings of the 2024 on Innovation and Technology in Computer Science Education V. 1, pp. 122–128. ITiCSE 2024, Association for Computing Machinery, New York, NY, USA (2024). https://doi.org/10.1145/3649217.3653533
2. Bhavya, B., Xiong, J., Zhai, C.: Analogy generation by prompting large language models: a case study of InstructGPT. In: Shaikh, S., Ferreira, T., Stent, A. (eds.) Proceedings of the 15th International Conference on Natural Language Generation. pp. 298–312. Association for Computational Linguistics, Waterville, Maine, USA and virtual meeting (2022). https://doi.org/10.18653/v1/2022.inlg-main.25, https://aclanthology.org/2022.inlg-main.25/

3. Bouchey, B., Castek, J., Thygeson, J.: Multimodal learning. In: Ryoo, J., Winkelmann, K. (eds.) Innovative Learning Environments in STEM Higher Education: Opportunities, Challenges, and Looking Forward, pp. 35–54. Springer International Publishing, Cham (2021). https://doi.org/10.1007/978-3-030-58948-6_3
4. Deriba, F.G., Sanusi, I.T., Sunday, A.O.: Enhancing computer programming education using ChatGPT- a mini review. In: Proceedings of the 23rd Koli Calling International Conference on Computing Education Research, pp. 1–2. Koli Calling '23, Association for Computing Machinery, New York, NY, USA (2024). https://doi.org/10.1145/3631802.3631848, https://dl.acm.org/doi/10.1145/3631802.3631848
5. Fincher, S., et al.: Notional machines in computing education: the education of attention. In: Proceedings of the Working Group Reports on Innovation and Technology in Computer Science Education, pp. 21–50. ACM, Trondheim Norway (2020). https://doi.org/10.1145/3437800.3439202, https://dl.acm.org/doi/10.1145/3437800.3439202
6. Gladys, M., Rogers, L., Sharafutdinova, G., Barnham, N., Nichols, P., Dastoor, P.C.: Driving course engagement through multimodal strategic technologies. Int. J. Innovation Sci. Math. Educ. **30**(3) (2022). https://doi.org/10.30722/IJISME.30.03.002, https://openjournals.library.sydney.edu.au/CAL/article/view/15793
7. Lakoff, G., Johnson, M.: Metaphors We Live By. University of Chicago Press, Chicago (1980)
8. Liffiton, M., Sheese, B.E., Savelka, J., Denny, P.: CodeHelp: using large language models with guardrails for scalable support in programming classes. In: Proceedings of the 23rd Koli Calling International Conference on Computing Education Research, pp. 1–11. Koli Calling '23, Association for Computing Machinery, New York, NY, USA (2024). https://doi.org/10.1145/3631802.3631830
9. Santos, I.M., Hauswirth, M., Jeuring, J.: Sound Notional Machines (2024). https://doi.org/10.31219/osf.io/z2vdr, https://osf.io/z2vdr_v1
10. Shutova, E.: Design and evaluation of metaphor processing systems. Comput. Linguist. **41**(4), 579–623 (2015). https://doi.org/10.1162/COLI_a_00233, https://aclanthology.org/J15-4002/, place: Cambridge, MA Publisher: MIT Press
11. Snelson, C.: Quest-based learning: a scoping review of the research literature. TechTrends **66**(2), 287–297 (2022). https://doi.org/10.1007/s11528-021-00674-w, https://link.springer.com/article/10.1007/s11528-021-00674-w, company: Springer Distributor: Springer Institution: Springer Label: Springer Number: 2 Publisher: Springer, US
12. The Manim Community Developers: Manim – Mathematical Animation Framework (2025). https://www.manim.community/, original-date: 2020-05-19T02:37:13Z
13. Tong, X., Choenni, R., Lewis, M., Shutova, E.: Metaphor understanding challenge dataset for LLMs. In: Ku, L.W., Martins, A., Srikumar, V. (eds.) Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pp. 3517–3536. Association for Computational Linguistics, Bangkok, Thailand (2024). https://doi.org/10.18653/v1/2024.acl-long.193, https://aclanthology.org/2024.acl-long.193/