Proceedings of Machine Learning Research 1:1–11, 2024 AAAI AI4Edu 24

Automated Assessment of Students' Code Comprehension using LLMs

Priti Oli POLI@MEMPHIS.EDU
Rabin Banjade RBNJADE1@MEMPHIS.EDU
Jeevan Chapagain JCHPGAIN@MEMPHIS.EDU
Vasile Rus VRUS@MEMPHIS.EDU
University of Memphis, Memphis TN 38152,USA

Abstract

Assessing students' answers and in particular natural language answers is a crucial challenge in the field of education. Advances in transformer-based models such as Large Language Models (LLMs), have led to significant progress in various natural language tasks. Nevertheless, amidst the growing trend of evaluating LLMs across diverse tasks, evaluating LLMs in the realm of automated answer assessment has not received much attention. To address this gap, we explore the potential of using LLMs for automated assessment of student's short and open-ended answers in program comprehension tasks. Particularly, we use LLMs to compare students' explanations with expert explanations in the context of line-by-line explanations of computer programs. For comparison purposes, we assess both decoder-only Large Language Models (LLMs) and encoder-based Semantic Textual Similarity (STS) models in the context of assessing the correctness of students' explanation of computer code. Our findings indicate that decoder-only LLMs, when prompted in few-shot and chain-of-thought setting perform comparable to fine-tuned encoder-based models in evaluating students' short answers in the programming domain.

Keywords: Automated Assessment, Large Language Model, Code Comprehension, Self-Explanation

1. Introduction

Large Language Models (LLMs), such as ChatGPT, have garnered significant attention for their remarkable ability to generate responses to user prompts. These models have been explored for their potential (and risks) for education, particularly in the realm of computer science (CS) education (Oli et al., 2023a), which is our focus. For CS education, use of LLMs in creating programming exercises (Sarsa et al., 2022), and generating code explanations (MacNeil et al., 2023) among other educational applications have been studied. While numerous studies have highlighted ChatGPT's generative capabilities of educational resources and assistance, there is a notable gap in exploring ChatGPT's assessment capabilities within educational contexts. In this work, we evaluate the effectiveness of LLMs to automatically assess students' self-explanations of code. Such explanations are generated, for instance, while students engage in code comprehension activities with a computer tutor, which needs to automatically assess the correctness of students' explanations of code to provide feedback. It should be noted that self-explanation, i.e., explaining learning material to oneself through speaking or writing (McNamara and Magliano, 2009), has been shown to improve comprehension and learning of programming concepts in introductory computer science courses (Tamang et al., 2021; Oli et al., 2023b). Additionally, prior studies have shown that the scaffolding of students' self-explanation is more effective than free self-explanation at improving novices' code comprehension (Oli et al., 2023b).

Scaffolding students' self-explanation relies on accurate assessment of students' explanations in terms of correctness and completeness. Manually assessing students' self-explanations and, consequently, their comprehension of the code is a challenging task for instructors, especially when dealing with large student cohorts. A simple and scalable approach to assessing student explanations is semantic similarity, i.e. measuring the similarity of such natural language explanations to an appropriate reference/correct answer, e.g., provided by an expert, through an automated short answer grading system (Mohler and Mihalcea, 2009). If the student's self-explanation is semantically similar to the reference answer the student's self-explanation is deemed to be correct.

Semantic similarity measures the degree to which two fragments of text have similar meanings by producing a similarity score, ranging from 0 to 1 (normalized score), 0 meaning no similarity at all, whereas 1 meaning semantically equivalent (Cer et al., 2017). Although there have been numerous studies (Cer et al., 2017) measuring semantic similarity between texts, limited research has been conducted in the area of computer programming and source code comprehension. In our study, we employ decoder-only Large Language Models to automatically evaluate students' line-by-line explanations of code and compare them with encoder-based models. We evaluate the proposed LLM-based approach using a set of student self-explanations produced in the context of an online learning environment that asks students to freely explain Java code examples line-by-line.

This work is part of a larger project whose primary objective is to create an educational technology that can scaffold students' understanding of code by providing tailored feedback to students while prompting students to explain their understanding of the code line-by-line. A key component of this system is assessing students' self-explanation of lines of code which we propose to do by computing the semantic similarity between each line of code and the corresponding student explanation.

2. Related Work

Automated Short Answer Grading (ASAG): Prior work on ASAG has been based on determining the semantic similarity between learner answers and reference answers in various domains such as Physics, Biology, Geometry etc. (Mohler and Mihalcea, 2009). The advances in neural networks led to the introduction of numerous deep learning-based Automated Short Answer Grading systems (Pontes et al., 2018) . Previous studies investigating pre-trained transformers in Natural Language Processing(NLP) tasks have observed significant performance improvements in automated short answer grading (Camus and Filighera, 2020) through fine-tuning on datasets such as MNLI (Williams et al., 2017) and SemEval-2013 (Segura-Bedmar et al., 2013). Sung et al. (2019) fine-tuned BERT on domain specific data such as textbooks and reported that fine-tuning a pre-trained model for task-specific purpose demonstrates superior performance in short answer grading. Along those lines, in our study, we fine-tune pre-trained models for assessing short answer in program comprehension, an area which has not been previously investigated.

Evaluating LLMs on Semantic Similarity Task: In their study, Zhong et al. (2023) report that ChatGPT surpasses all BERT models with a substantial margin in an inference task and attains comparable performance to BERT in sentiment analysis and question-answering tasks. However, their study indicates that ChatGPT has limited ability in paraphrase and semantic similarity tasks. However, Gatto et al. (2023) demonstrate that the Semantic Textual Similarity (STS) task can be effectively framed as a text generation problem, achieving robust performance with LLM outperforming encoder-based STS models across various STS benchmarks.Given that LLMs benefit significantly from training on code and its corresponding summaries, in this study we investigate the applicability of LLMs to automatically assess students' line-by-line explanations of code.

3. Dataset

In our work, we use the *SelfCode* (Chapagain et al., 2023) for our analysis which consists of pairs of code snippets accompanied by expert explanations and explanations given by students/non-experts for ten different code examples. To collect the student's explanations, Amazon Mechanical Turk (MTurk) was used with the MTurk HIT (Human Intelligence Task) being available only to workers from the United States and Canada who had to qualify for the task by correctly answering 2 out of 3 multiple choice basic program construction tasks. Expert explanations were acquired from a curated collection of annotated examples within a comprehensive repository of interactive learning content (Hicks et al., 2020). These expert explanations serve as reference explanations when assessing learners' self-explanations.

In addition to expert explanations, human judgments of the semantic similarity between the expert and students' code explanations were obtained. Six graduate students in Computer Science annotated on a scale of 1-5 about 1,770 pairs of expert and student's explanations which are used in our study presented here. Before beginning the annotations, the graduate students received training on the annotation guidelines. The annotation occurred in multiple stages: the first 100 data instances were used to establish a consistent understanding of the annotation process. In the subsequent steps annotators involved a disagreement mitigation step aiming to minimize score differences to within 1 point among annotators and the inter-annotator agreement (Fleiss, 1971) was computed to be 0.99 indicating high agreement among annotators.

In the data set, 18% of the sentence pairs scored 4 or 5 (high semantic similarity), while 59% were labeled incorrect (score 1) or exhibited low concept coverage (score 2). About 23% of the sentence pairs received a score of 3. Given the opaque nature of ChatGPT's training data, we validate our findings against memorization by exclusively working with publicly released datasets after May 2023.

4. Methodology

As already noted, we employ semantic similarity to evaluate students' natural language responses, with the primary focus on assessing decoder-only LLMs' capability in measuring semantic similarity; however, for comparison purposes, we offer results with several other approaches, as described next.

4.1. Assessment Using Encoder Models

First, we calculate the similarity based on BERTScore (Zhang et al., 2019) and Universal Sentence Encoder (USE; (Cer et al., 2018)). Second, we employ Sentence transformer models (Reimers and Gurevych, 2019b), which we fine-tune on our dataset. The three pretrained sentence transformer models that we further fine-tune with our dataset include: i) SRoBERTa fine-tuned on NLI, ii) CodeBERT, and iii) all-mpnet-base-v2[1]. We experimented with CodeBERT (Feng et al., 2020) as an encoder to assess whether it offers advantages in capturing the similarity of sentences related to code segments. Additionally, sentence transformer models demonstrate improved performance in tasks related to Natural Language Inference (NLI) when fine-tuned on models previously trained with NLI data (Reimers and Gurevych, 2019a). Hence, we fine-tuned models that were initially trained in NLI using our dataset for enhanced performance.

1. https://huggingface.co/sentence-transformers/all-mpnet-base-v2

For each of the mentioned encoders, we compute the similarity between expert and student explanations by calculating the cosine similarity between their embeddings. There is an exception to this similarity computation when calculating BERTScore. In this case, the similarity of two sentences is computed as the sum of cosine similarities between their tokens' embeddings.

We split our data-set into 80% training data and 20% test data and fine-tuned SBERT with contrastive loss objective function for one epoch in our training dataset. We used a batch size of 16, Adam optimizer with a learning rate $2e^{-4}$ and a linear learning rate warm-up over 10% of the training data. Our pooling strategy is MEAN. This comprehensive assessment framework allows us to thoroughly evaluate the effectiveness of different language models and baselines in capturing semantic similarity in the context of answer assessment.

4.2. Assessment by Prompting LLMs:

We explore various prompting strategies for four different large language models: OpenAI's ChatGPT-3.5-turbo-0613 and ChatGPT-4-0613 (OpenAI, 2023), gpt-4-1106-preview(GPT-4 Turbo) (OpenAI, 2023) and Meta's open source model LLMa2-chat[2] (Touvron et al., 2023)).

First, for predictive prompting of semantic similarity, we used simple prompts to instruct the LLM to predict the similarity score on a scale of 1-5, similar to human judgments (with 1 indicating no semantic similarity and 5 indicating semantically equivalence between the pair of sentences). Based on the findings by Gatto et al. (2023), who suggest framing STS tasks to predict a similarity percentage (leveraging large language models' strong textual reasoning and their exposure to percentage-related language during pre-training), we further used the same prompt to generate the semantic similarity in the scale of 0-1. In addition, we also explore advanced prompting strategies. These include the conventional few-shot prompting, also known as in-context learning, where the LLM is tasked to infer from the provided examples or task descriptions (Brown et al., 2020), as well as few-shot chain-of-thought (CoT) prompting (Wei et al., 2022) where the LLM is guided to think step by step. In the case of few-shot learning, we employed a stratified sampling approach to select six expert explanations along with corresponding student explanations and benchmark similarity scores. These were provided as examples to the Large Language Models (LLMs), with the caveat that the examples were excluded from the dataset used for subsequent analysis.

For few-shot Chain-of-Thought prompting, we manually crafted a step-by-step breakdown of the reasoning behind assigning semantic similarity scores when evaluating two texts, selecting three examples with varying benchmark similarity scores. The prompts utilized in our analysis are detailed in Appendix B. In the CoT Prompting approach, which elicited textual responses along with reasoning, we extracted numerical values within specified delimiters to obtain the semantic similarity score. In our experimental setup, we opted for deterministic results by setting the temperature parameter to 0. We set a maximum token length of 1200 to limit the scope of generated sequences.

5. Results and Discussion

5.1. Assessment using Encoder-based Models

As we can see from results in Table 1, for encoder based models, models such as BERTScore and Universal Sentence Encoder show below par results based on Pearson and Spearman rank correlation. The results indicate that fine-tuned sentence transformer models capture the assessment

2. https://huggingface.co/meta-llama/Llama-2-7b-chat-hf

	Model	Pearson	Spearman
	BERTScore	0.573	0.553
	USE	0.61	0.61
Sentence Transformer	RoBERTa-base†	0.800	0.78
	CodeBERT-base†	0.797	0.761
	all-mpnet†	**0.81**	**0.811**
GPT-3.5	baseline-prompt[1-5]	0.58	0.59
	baseline-prompt[0-1]	0.60	0.61
	fewshot-prompt[0-1]	0.64	0.64
	CoT-prompt[0-1]	0.69	0.70
GPT-4	baseline-prompt[1-5]	0.69	0.70
	baseline-prompt[0-1]	0.72	0.737
	fewshot-prompt[0-1]	0.78	0.79
	CoT-prompt[0-1]	**0.81**	**0.82**
GPT-4-turbo	baseline-prompt[1-5]	0.70	0.70
	baseline-prompt[0-1]	0.72	0.75
	fewshot-prompt[0-1]	0.67	0.71
	CoT-prompt[0-1]	0.79	0.80
LLAMA-2	baseline-prompt[1-5]	0.29	0.31
	baseline-prompt[0-1]	0.38	0.39
	few-shot-prompt[0-1]	0.42	0.44
	CoT-prompt[0-1]	0.26	0.27

Table 1: Pearson and Spearman correlations by comparing human-annotated semantic similarity scores with automated similarity scores for student and expert explanations across different model classes. † indicate fine-tuned model

score better. The encoder models pre-trained on code such as CodeBERT do not show better performance compared to RoBERTa. The best performing model for student answer explanation is *all-mpnet*. One of the reasons for this might be the large amount of data it is fine-tuned on. Also, there is no remarkable difference between RoBERTa and *all-mpnet* indicating sentence transformer models can be used effectively for student answer assessment by comparing expert explanations with student explanations.

5.2. Assessment by Prompting Decoder-only LLMs

In Table 1, we present results of prompting decoder-only Large Language Models (LLM) to assess semantic similarity. The outcomes for various versions of ChatGPT indicate a notable trend: prompting the LLMs to predict semantic similarity on a scale of 0-1 yields superior performance compared to prompting it to predict similarity on other arbitrary scales (1-5).

The advance strategies consistently boost ChatGPT's performance, with manual chain-of-thought (CoT) providing the most significant benefits. Notably, the standard few-shot CoT enhances ChatGPT's overall performance (on average 15% better than baseline prompting for ChatGPT-based model) with GPT-4 providing the best performance for our task. Table 1 shows that GPT4 performs similarly to fine-tuned encoder-based models when using chain-of-thought prompting. The results also indicate that GPT-4 consistently outperforms GPT-3.5 across various prompting techniques and scales. Our experimentation with GPT-4-turbo yielded results comparable to those of other LLMs, offering no discernible advantage except processing speed. ChatGPT-4 demonstrates superior reasoning in CoT-prompting and also closely aligns with human-annotated benchmark similarity (see Appendix C.1 for examples). In the case of LLama-2, the semantic similarity scores were skewed towards higher values, particularly with scores of 0.8-1.0 in the scale of 0-1. Moreover, we observed that LLama-2 generates verbose results, including reasoning about the semantic similarity score often deviating from instruction prompt provided.

5.3. Error Analysis

When prompting LLMs, we found in-context learning to be sensitive to the provided examples, which is consistent with the findings from previous studies (Agrawal et al., 2022; Zhong et al., 2023). This sensitivity may arise from limited generalization or overfitting to few-shot examples used, suggesting a potential correlation between provided examples and test data. To address this potential bias in the few-shot setting, we conducted the analysis three times with different instances of example provided each time and present the results as the average of these runs. One of the cases where the LLMs fail is for instances involving numerical reasoning. LLMs assign a high semantic equivalence score to instances, which although linguistically similar, involve different numerical values.

6. Conclusion

This work investigated the ability of LLMs to automatically assess students' code comprehension. Our results indicate that Large Language Models (LLMs) perform comparably well, in particular GPT models, to fine-tuned encoder-based models but there is room for improvement which we plan to explore in the future.

Acknowledgments

This work has been supported by the following awards: NSF #1934745, #1918751, #1822816, and IES award R305A220385. The opinions, findings, and results are solely those of the authors and do not reflect those of NSF or IES.

References

Sweta Agrawal, Chunting Zhou, Mike Lewis, Luke Zettlemoyer, and Marjan Ghazvininejad. In-context examples selection for machine translation. *arXiv preprint arXiv:2212.02437*, 2022.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

Leon Camus and Anna Filighera. Investigating transformers for automatic short answer grading. In *Artificial Intelligence in Education: 21st International Conference, AIED 2020, Ifrane, Morocco, July 6–10, 2020, Proceedings, Part II 21*, pages 43–48. Springer, 2020.

Daniel Cer, Mona Diab, Eneko Agirre, Inigo Lopez-Gazpio, and Lucia Specia. Semeval-2017 task 1: Semantic textual similarity-multilingual and cross-lingual focused evaluation. *arXiv preprint arXiv:1708.00055*, 2017.

Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, et al. Universal sentence encoder. *arXiv preprint arXiv:1803.11175*, 2018.

Jeevan Chapagain, Zak Risha, Rabin Banjade, Priti Oli, Lasang Tamang, Peter Brusilovsky, and Vasile RUs. Selfcode: An annotated corpus and a model for automated assessment of self-explanation during source code comprehension. In *The International FLAIRS Conference Proceedings*, volume 36, 2023.

Zhangyin Feng, Daya Guo, Duyu Tang, Nan Duan, Xiaocheng Feng, Ming Gong, Linjun Shou, Bing Qin, Ting Liu, Daxin Jiang, et al. Codebert: A pre-trained model for programming and natural languages. *arXiv preprint arXiv:2002.08155*, 2020.

Joseph L Fleiss. Measuring nominal scale agreement among many raters. *Psychological Bulletin*, 76 (5):378, 1971.

Joseph Gatto, Omar Sharif, Parker Seegmiller, Philip Bohlman, and Sarah Masud Preum. Text encoders lack knowledge: Leveraging generative llms for domain-specific semantic textual similarity. *arXiv preprint arXiv:2309.06541*, 2023.

Alexander Hicks, Kamil Akhuseyinoglu, Clifford Shaffer, and Peter Brusilovsky. Live catalog of smart learning objects for computer science education. In *Sixth SPLICE Workshop*, 2020.

Stephen MacNeil, Andrew Tran, Arto Hellas, Joanne Kim, Sami Sarsa, Paul Denny, Seth Bernstein, and Juho Leinonen. Experiences from using code explanations generated by large language models in a web software development e-book. In *Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1*, pages 931–937, 2023.

Danielle S McNamara and Joseph P Magliano. Self-explanation and metacognition: The dynamics of reading. In *Handbook of metacognition in education*, pages 60–81. Routledge, 2009.

Michael Mohler and Rada Mihalcea. Text-to-text semantic similarity for automatic short answer grading. In *Proceedings of the 12th Conference of the European Chapter of the ACL (EACL 2009)*, pages 567–575, 2009.

Priti Oli, Rabin Banjade, Jeevan Chapagain, and Vasile Rus. The behavior of large language models when prompted to generate code explanations. *arXiv preprint arXiv:2311.01490*, 2023a.

Priti Oli, Rabin Banjade, Arun Balajiee Lekshmi Narayanan, Jeevan Chapagain, Lasang Jimba Tamang, Peter Brusilovsky, and Vasile Rus. Improving code comprehension through scaffolded self-explanations. In *International Conference on Artificial Intelligence in Education*, pages 478–483. Springer, 2023b.

OpenAI. Gpt-4 technical report. *CoRR*, abs/2303.08774, 2023.

Elvys Linhares Pontes, Stéphane Huet, Andréa Carneiro Linhares, and Juan-Manuel Torres-Moreno. Predicting the semantic textual similarity with siamese cnn and lstm. *arXiv preprint arXiv:1810.10641*, 2018.

Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 11 2019a. URL http://arxiv.org/abs/1908.10084.

Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084*, 2019b.

Sami Sarsa, Paul Denny, Arto Hellas, and Juho Leinonen. 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*, pages 27–43, 2022.

Isabel Segura-Bedmar, Paloma Martínez Fernández, and María Herrero Zazo. Semeval-2013 task 9: Extraction of drug-drug interactions from biomedical texts (ddiextraction 2013). Association for Computational Linguistics, 2013.

Chul Sung, Tejas Dhamecha, Swarnadeep Saha, Tengfei Ma, Vinay Reddy, and Rishi Arora. Pre-training bert on domain resources for short answer grading. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 6071–6075, 2019.

Lasang Jimba Tamang, Zeyad Alshaikh, Nisrine Ait Khayi, Priti Oli, and Vasile Rus. A comparative study of free self-explanations and socratic tutoring explanations for source code comprehension. In *Proceedings of the 52nd ACM Technical Symposium on Computer Science Education*, pages 219–225, 2021.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, 35:24824–24837, 2022.

Adina Williams, Nikita Nangia, and Samuel R Bowman. A broad-coverage challenge corpus for sentence understanding through inference. *arXiv preprint arXiv:1704.05426*, 2017.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q Weinberger, and Yoav Artzi. Bertscore: Evaluating text generation with bert. *arXiv preprint arXiv:1904.09675*, 2019.

Qihuang Zhong, Liang Ding, Juhua Liu, Bo Du, and Dacheng Tao. Can chatgpt understand too? a comparative study on chatgpt and fine-tuned bert. *arXiv preprint arXiv:2302.10198*, 2023.

Appendix A. Dataset Distribution

Similarity Score (1-5)	No. of Sentence Pair
1	529 (29.88%)
2	507 (28.62%)
3	419 (23.65%)
4	253 (14.45%)
5	62 (3.50%)

Table 2: Distribution of Data

Appendix B. Prompts

Baseline Prompt [1-5]: Analyze if the two sentences are similar and provide a score between 1 to 5, with 1 indicating minimal similarity and 5 representing maximal similarity. Provide semantic similarity score for <*user explanation*> and <*expert explanation*> between 1 to 5. Only provide the score without any other text.

Baseline Prompt [0-1]: Assess the similarity of the two sentences and assign a similarity score on a scale from 0 to 1, with 0 indicating minimal similarity and 1 representing maximal similarity. Provide semantic similarity score for <*user explanation*> and <*expert explanation*> between 0 to 1. Only provide the score without any other text.

Few Shot Prompt [0-1]: Assess the similarity of the two sentences and assign a similarity score on a scale from 0 to 1, with 0 indicating minimal similarity and 1 representing maximal similarity. Provide a semantic similarity score for 'Declares the array we want to use for our assignment' and 'We initialize the array of type int to hold the specified numbers.' between 0 and 1. Only provide the score without any other text. Similarity Score: 0.87

Assess the similarity of the two sentences and assign a similarity score on a scale from 0 to 1, with 0 indicating minimal similarity and 1 representing maximal similarity. Provide semantic similarity score for 'run a while-loop as long as the remainder of num/divisor is not equal to 0' and 'We could check whether the divisor is not a factor of the number by computing the remainder of the division of the number by the divisor.' between 0 and 1. Only provide the score without any other text. Similarity Score: 0.75
....
Assess the similarity of the two sentences and assign a similarity score on a scale from 0 to 1, with 0 indicating minimal similarity and 1 representing maximal similarity. Provide semantic similarity score for <*user explanation*> and <*expert explanation*> between 0 to 1. Only provide the score without any other text. Similarity Score:

Chain-of-Thought (CoT) Prompt [0-1]: Discuss how these two texts are similar and different, then assign a semantic similarity score between [0.0-1.0] which describes their semantic similarity: ‘Declares the array we want to use for our assignment, and ‘We initialize the array of type int to hold the specified numbers. Similarity: Lets think step by step. Both the text is about declaration or initialization of array. The slight difference between the two texts is the second text provides additional information about the type of the declared array. Thus, these sentences have a [semantic similarity = 0.87]

Discuss how these two texts are similar and different, then assign a semantic similarity score between [0.0-1.0] which describes their semantic similarity: ‘We could check whether the divisor is not a factor of the number by computing the remainder of the division of the number by the divisor.’ and ‘run a while-loop as long as the remainder of num/divisor is not equal to 0’ Similarity: Lets think step by step. Both the text is about computing the checking whether divisor is a factor of number or not. However, the first text is more specific about using a while-loop and the condition for the loop to continue, while the second text is more focused on the purpose of the operation, which is to check if the divisor is a factor of the number. Thus, these sentences have a [semantic similarity = 0.75]

Discuss how these two texts are similar and different, then assign a semantic similarity score between [0.0-1.0] which describes their semantic similarity: ‘Loop start’ and ‘We need to increment the divisor repeatedly as long as the divisor is not a factor of the number.Similarity: Lets think step by step. Both texts are discussing loop. The first text is simply stating the start of a loop, while the second text is explaining a specific condition within a loop. Thus, these sentences have a [semantic similarity = 0.2]

Discuss how these two texts are similar and different, then assign a semantic similarity score between [0.0-1.0] which describes their semantic similarity:<*user explanation*> and <*expert explanation*> Similarity: Lets think step by step

Appendix C. Chain-of-Though-Prompting Example

C.1. LLM response with semantic similarity and reasoning for CoT Prompting

Code statement: System.out.println("The integer is positive.");
Expert explanation: Print that the integer is positive if it is greater than 0.
Student explanation: This statement prints that the integer is positive.
Benchmark Similarity: 1
LLM: GPT-4
Both texts are discussing the same action of printing a statement that an integer is positive. The first text, however, provides a condition for this action (if the integer is greater than 0), while the second text does not specify any condition. Thus, these sentences have a [semantic similarity = 0.8]

Code statement: int divisor = 2;
Expert explanation: We define variable divisor to store the smallest divisor of the number.
Student explanation: we initialize another variable named divisor with the value 2.
Benchmark Similarity: 0.33
LLM: GPT-4
Both texts discuss initializing a variable named 'divisor', but they differ in the context and purpose of this variable. The first text simply states that the variable is initialized with the value 2, while the second text explains that the variable is used to store the smallest divisor of a number. Thus, these sentences have a [semantic similarity = 0.6]

C.2. Fail case: LLM fails in numerical reasoning

Code statement: int num = 15;
Expert explanation: In this program, we initialize variable num to 15.
Student explanation: creates variable integer entitled "num" with initial value 5
Benchmark Similarity: 0.466666667
LLM: GPT-4
LLM response to Chain-of-thought Prompting: Both texts are discussing the initialization of a variable named "num", but the initial values they assign to "num" are different. The first text assigns a value of 5, while the second assigns a value of 15. Thus, these sentences have a [semantic similarity = 0.8]
Comment: LLM assigns high semantic equivalence score to the two text, which although linguistically similar involve different numerical value. In such situations where there is a numerical disparity between a student's explanation and an expert's explanation, current Language Models (LLMs) do not account for this difference when automatically evaluating the similarity between the two texts.