Summarising and Analysing Research Papers

Assignment

BHASKAR JYOTI KAUSHIK

2024

**Topic Selection:**

The chosen topic from the research paper **"Leveraging GPT and Ordered Networks for Understanding Introductory Programming Education”**.

**Research Paper Reference:**

Pankiewicz, M., Zambrano, A. F., Barany, A., & Baker, R. S. (2024). **How we Code Code: Leveraging GPT and Ordered Networks for Understanding Introductory Programming Education**. Available at: [**ResearchGate Link**](https://www.researchgate.net/publication/383415768_How_we_Code_Code_Leveraging_GPT_and_Ordered_Networks_for_Understanding_Introductory_Programming_Education/download?_tp=eyJjb250ZXh0Ijp7ImZpcnN0UGFnZSI6ImluZGV4IiwicGFnZSI6InNjaWVuY2VUb3BpYyIsInByZXZpb3VzUGFnZSI6ImluZGV4In19) :<https://www.researchgate.net/publication/383415768>

**Initial Prompt for Summary:**

"Summarize the key findings and methodologies of the research paper 'How we Code Code' that focuses on leveraging GPT and Ordered Networks to understand programming education."

**Generated Summary (Iteration 1):**

This paper investigates the use of GPT-based models and Ordered Network Analysis (ONA) to analyze debugging behaviors in novice programmers. The study explores data from an automated programming assessment system used in a university-level CS1 course. It identifies patterns in how high- and low-performing students address compiler errors and failed unit tests. Using a large dataset of student code submissions, the researchers applied qualitative coding and ONA to highlight differences in debugging strategies. The study finds that high-performing students tend to focus on key concepts like conditional statements and return values, while low-performing students make less targeted changes, often addressing irrelevant sections of the code.

**Prompt for Insights Extraction:**

"Based on the results, what key insights can we draw about the learning behaviors of high- and low-performing novice programmers, and how can these findings inform teaching strategies?"

**Extracted Insights (Iteration 1):**

The analysis reveals that high-performing students are more focused on debugging critical parts of the code such as conditional statements, while low-performing students often modify unrelated sections like the testing area or add/remove lines unnecessarily. These findings suggest that successful students understand the specific sources of errors, while less successful students exhibit a trial-and-error approach. This highlights the need for teaching strategies that help low-performing students develop better debugging practices early on. Instructors could use targeted interventions to guide students in focusing on relevant parts of the code when solving compiler errors or failed tests.

**Prompt for Application Suggestions**:

"Suggest practical applications or educational interventions that can be derived from these findings to improve student outcomes in programming courses."

**Suggested Applications:**

1. **Automated Feedback Systems**: Develop adaptive feedback tools that use insights from this research to provide personalized guidance, focusing students on specific areas of code related to their errors.

2. **Early Diagnostic Tools**: Implement assessments in the early weeks of programming courses that identify students who are prone to trial-and-error debugging and offer targeted lessons on debugging best practices.

3. **Interactive Learning Modules**: Create modules that teach students how to diagnose and address specific types of errors, such as compiler versus logic errors, using examples from high-performing students.

**Evaluation**:

The summaries and insights generated provide clear and relevant information about the research paper. The key findings are well-represented, and the insights into student learning behaviors are accurate. The applications proposed are practical and align with the research's emphasis on targeted debugging interventions. Overall, the generated content meets clarity and relevance standards.

**Reflection**:

Working through this exercise offered valuable insights into the application of qualitative coding and ONA in understanding student behavior. The use of GPT to automate qualitative analysis is particularly promising for large-scale educational research. A challenge I encountered was ensuring that the generated summaries remained focused on the paper's key contributions without becoming too general. Through iteration, I learned to refine the prompts for better specificity, ultimately enhancing the quality of the summaries and insights.