ANLY 725: Prompt Engineering Experiment Proposal

**Bad at Math and Coding?...**

**Evaluating the Performance of Large Language Models in Answering Math Questions and Generating Programming Code**

Melanie Cirici, Darren Stewart

October 14, 2024

#### ***Introduction***

Large language models like ChatGPT are celebrated for their prowess in creating coherent text and even engaging in creative writing. However, they often struggle when it comes to solving mathematical problems accurately.1,2 Our objective is to investigate whether the format of mathematical prompts—expressed in either numeric/symbolic formulas or in natural language—affects the LLM's ability to solve them correctly. Moreover, we aim to extend this evaluation to understand how LLMs generate code in programming languages like Python and R, focusing on problem-solving accuracy and efficiency.3

#### ***Research Hypotheses***

We hypothesize the following:

1. **Language Dependency Hypothesis:** LLMs will perform better when math problems are expressed predominantly in English words rather than in numeric or symbolic formats.
2. **Complexity and Task Variation Hypothesis:** The accuracy of LLMs will significantly vary with the type and complexity of the math problem.
3. **Consistency Hypothesis:** LLMs may produce different answers for the same complex prompt when temperature settings vary, leading to inconsistencies.
4. **Programming Task Hypothesis:** For code generation tasks, LLMs will yield higher accuracy with more detailed, natural language prompts than with technical, terse commands.

#### ***Research Design***

We will conduct our experiments on a range of LLMs,4 including GPT-3.5, GPT-4, Claude, and others. The specific set of LLMs tested will not exceed 4 and be selected from among the latest publicly available LLMs we are aware of at the time of the study. A similar study design will be used for the two sub-studies contained in this proposal: (1) evaluating math problem-solving and (2) code generation capabilities. Each prompt type will be tested under different conditions, using a controlled set of metrics to measure accuracy, response time, and consistency.

#### ***Study Design Structure***

##### **(Sub-study 1) Math Problem Evaluation:** Example prompts are shown in Table 1. The final set of prompts will include at least 2 (simple and complex) from each category, or type, of math problem.

***Table 1: Math Problem Prompt Engineering Study Design Example***

| **Model** | **IV1: Mode of Expression** | **IV2: Type of Math Problem** | **IV3: Complexity of Math Problem** | **DV** | **Prompt Example** | **Correct Answer** | **LLM Response** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| LLM 1 | Numeric/symbolic | Arithmetic | Simple | Accuracy | What is 6+9+10? | 25 | LLM-generated |
| LLM 1 | English words | Arithmetic | Simple | Accuracy | What is six plus nine plus ten? | 25 | LLM-generated |
| LLM 2 | Numeric/symbolic | Arithmetic | Complex | Accuracy | What is 6+9+99999+4.5-81.2-0+10? | 99947.3 | LLM-generated |
| LLM 2 | English words | Arithmetic | Complex | Accuracy | What is six plus nine plus ninety-nine thousand, nine-hundred and ninety-nine plus four and a half...? | 99947.3 | LLM-generated |
| LLM 3 | Numeric/symbolic | Multiplication | Simple | Accuracy | What is 8 \* 7? | 56 | LLM-generated |
| LLM 3 | English words | Multiplication | Simple | Accuracy | What is eight times seven? | 56 | LLM-generated |
| LLM 3 | Numeric/symbolic | Algebra | Intermediate | Accuracy | Solve for x if 2x + 5 = 9. | x = 2 | LLM-generated |
| LLM 3 | English words | Algebra | Intermediate | Accuracy | Solve for x if two times x plus five equals nine. | x = 2 | LLM-generated |

We will implement a structured evaluation of multiple types of math problems, including arithmetic, multiplication, division, algebra, and statistics using different formats (numeric/symbolic and English words). The task complexity ranged from simple to intermediate to more complex, ensuring that a diverse set of mathematical challenges was covered.

To ensure precise measurement, we will develop a Python code from – ***Tables 1 and 2*** – to simulate LLM responses and evaluate their accuracy based on predefined mock answers. This method allowed us to analyze response time and consistency effectively.

##### **(Sub-study 2) Code Generation Evaluation:**

***Table 2: Code Generation Prompt Engineering Study Design Example***

| **Model** | **IV1: Mode of Expression** | **IV2: Type of Code Task** | **IV3: Complexity of Code Task** | **DV** | **Prompt Example** | **Expected Output** | **LLM Response** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| LLM 1 | Technical Language | Data Analysis (Python) | Simple | Accuracy | Write Python code to find the mean of a list. | Mean of numbers | LLM-generated |
| LLM 1 | Natural Language | Data Analysis (Python) | Simple | Accuracy | Write a Python code snippet to calculate the mean of a list. | Mean of numbers | LLM-generated |
| LLM 2 | Technical Language | Statistical Analysis (R) | Complex | Accuracy | Generate R code for linear regression. | Regression coefficients | LLM-generated |
| LLM 2 | Natural Language | Statistical Analysis (R) | Complex | Accuracy | Generate R code to perform a linear regression without lm. | Regression coefficients | LLM-generated |
| LLM 3 | Technical Language | Data Visualization (Python) | Simple | Accuracy | Create a bar chart in Python. | ASCII representation | LLM-generated |
| LLM 3 | Natural Language | Data Visualization (Python) | Simple | Accuracy | Write Python code to create a bar chart without using libraries. | ASCII representation | LLM-generated |

#### **Metrics for Evaluation**

* **Accuracy:** We will measure the correctness of the responses for both math problems and code generation.
* **Response Time:** Time taken by the LLM to generate the answer or code.
* **Consistency:** Testing if multiple runs of the same prompt yield the same response.
* **Error Rate:** Number of logical or syntax errors in the generated code.

#### **Analysis Plan**

We will use statistical methods to analyze the results:

* **Accuracy Analysis:** To compare prompt formats and how well different LLMs handle numeric/symbolic expressions versus English words.
* **Response Time Analysis:** To assess the speed of responses for each model, broken down by prompt type and complexity.
* **Consistency Analysis:** Testing each prompt multiple times to see how often the results are consistent, especially with higher temperature settings.

#### **Conclusion**

This study aims to provide insights into how prompt engineering affects LLM performance in both math problem-solving and code generation. By systematically evaluating LLMs under different conditions, we hope to identify the most effective prompt structures for increasing accuracy and consistency.

#### **References**

1. Ahn, J., Verma, R., Lou, R., Liu, D., Zhang, R., & Yin, W. (2024). Large language models for mathematical reasoning: Progresses and challenges. *arXiv preprint arXiv:2402.00157*.
2. Zhang, Y., Xue, M., Liu, D., & He, Z. (2024, August). Rationales for Answers to Simple Math Word Problems Confuse Large Language Models. In *Findings of the Association for Computational Linguistics ACL 2024* (pp. 8853-8869).
3. Liu, J., Xia, C. S., Wang, Y., & Zhang, L. (2024). Is your code generated by chatgpt really correct? rigorous evaluation of large language models for code generation. *Advances in Neural Information Processing Systems*, *36*.
4. Naveed, Humza, Asad Ullah Khan, Shi Qiu, Muhammad Saqib, Saeed Anwar, Muhammad Usman, Naveed Akhtar, Nick Barnes, and Ajmal Mian. "A comprehensive overview of large language models." *arXiv preprint arXiv:2307.06435* (2023).

#### **Contributions**

Darren: Conceptualized study design; drafted initial study plan; added some citations.  
  
Melanie: Revised draft and added sub-study 2, citation and Tables.