Cognicue

Problem Statement

What?

- Interactions with Large Language Models (LLMs) in coding tasks can increase cognitive load, particularly as task complexity and LLM usage increase
- The goal is to identify and recommend ways to optimize LLM interactions to prevent cognitive overload

Why?

- LLMs should be able to assist with coding, but if the interactions lead to cognitive overload, they can negatively impact the user's ability to focus, problem-solve, and maintain efficiency.
- This can hinder the long-term usability and effectiveness of LLMs in coding environments, which defeats their purpose of enhancing productivity.

prompts [117, 125]. However, recent research has indicated that crafting effective prompts is challenging [122], particularly for those without a deep understanding of AI [5, 43, 122]. The conversational interface of these chatbots mimics human interaction [101], potentially misleading students into thinking that prompt-writing is as straightforward as talking to humans [53]. This gap between perceived simplicity and the actual complexity of effective prompt-writing can breed overconfidence. However, when the chatbot fails to respond as expected, students' trust in the AI system [32] can wane, leading to frustration [88]. In response to these challenges, informal

Impact of Guidance and interaction Strategies for LLM use on learner performance and perception.
Proc. ACM Hum.-Comput. Interact., Vol. 8.

Applications & Impact

1

Enhanced Developer Productivity

LLMs streamline coding by simplifying complex tasks, offering intelligent code suggestions, and minimizing context switching, allowing developers to focus on high-level problem-solving.

2

Democratization of Software Development

Optimized LLM prompts enable non-technical users to create programs through natural language, making software development more accessible via low-code/no-code platforms.

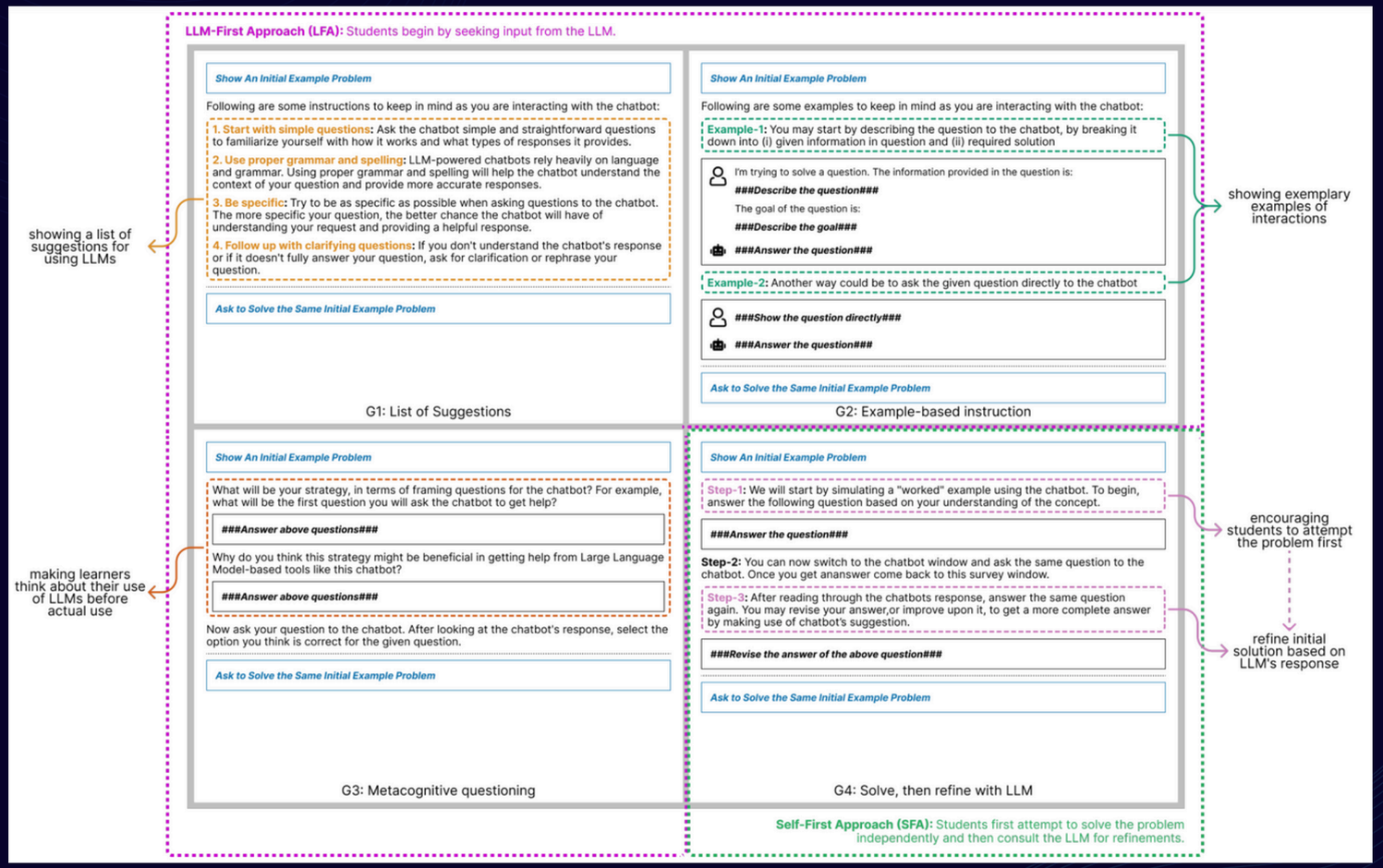
3

Accelerated Innovation & Prototyping

LLMs reduce cognitive load, enabling rapid idea testing, technology experimentation, and solution prototyping.

Literature Review

Impact of Guidance and Interaction Strategies for LLM Use on Learner Performance and Perception



- Assigned students to 1 of 4 treatment groups (guidance strategies) as shown in the image
- Categorised their 1st prompts into 4 categories such as unrelated prompt, exact prompt from the question etc.
- Compared performance between treatment and control groups
- Found significant improvement in accuracy with guidance strategies

Harsh Kumar, Iya Musabirov et. al.
University of Toronto, Canada
20th August, 2024

A Study on Developer Behaviors for Validating and Repairing LLM-Generated Code Using Eye Tracking and IDE Actions

- Investigated use of Co-pilot for code generation and correction
- Tasks - Dynamic programming algorithm, calculator and zoo management system
- 1 group was informed code is generated by Copilot and 1 group was not
- Participants were more likely to delete and rewrite code by themselves when informed it was generated by Copilot, increasing cognitive load
- Exhibit higher saccade times, used the clipboard less frequently, experience a higher cognitive workload, as indicated by self-reported effort, fixation time, and average fixation duration

NASA TLX

After completing each task, we asked participants to complete a NASA Task Load Index (NASA-TLX) questionnaire [32] to self-report their cognitive workload. NASA-TLX is a widely-used subjective assessment tool that measures the user's perceived workload when performing a task. It includes six dimensions: Mental Demand, Physical Demand, Temporal Demand, Performance, Effort, and Frustration. We used it to complement the data obtained from eye tracking.

CodeGRITS

To support the analysis of developer behaviors, we used CodeGRITS [31], a plug-in for IntelliJ IDEA that enabled IDE tracking and eye tracking. In addition, we recorded the

Ningzhi Tang, Meng Chen et. al.
University of Notre Dame
25th May, 2024

Prompt engineering in consistency and reliability with the evidence-based guideline for LLMs

- The authors applied various prompt types to explore how different prompt designs impact LLM performance
- Input-Output (IO) Prompting, Chain of Thought (COT) Prompting, Reflection of Thoughts (ROT) Prompting
- ROT prompting showed the highest consistency rates, (77.5%)
- While IO prompting in some gpt-3.5 models achieved near-perfect reliability, other prompts like COT and ROT varied across different models
- ROT prompting, in particular, has the potential to guide LLMs in providing more accurate and thoughtful responses

Prompt	Definition
Input-output (IO) prompting	Input the instruction directly
0-shot-Chain of thought (0-COT) prompting	Use “Think it step by step” on the base of IO to steer the LLM complete reasoning.
Performed-Chain of thought (P-COT) prompting	Break down the task into different steps to perform what reasoning processes need to be conducted by the LLM.
Reflection of thoughts (ROT) prompting	Break down the task into different steps and steer the LLM to backtrack previous steps by let the LLM simulates the mode of discussion.

Wang, L., Chen, X., Deng, X. et al
20th February, 2024

Gaps identified

Paper 1

- Focused on students studying computer science - we will take a range of majors and people with different coding experience levels
- Cognitive load and user experience is not considered, only accuracy of responses and final result is measured

Paper 2

- Again, focused on students studying computer science
- Just studies impact of copilot use on cognitive load, does not suggest any strategies to better interact with LLMs

Paper 3

- Focused on medical diagnosis and information
- Implication for future scope - do the results of our analysis apply to different LLMs?

Experimental Design

Experimental Design

01 Participants

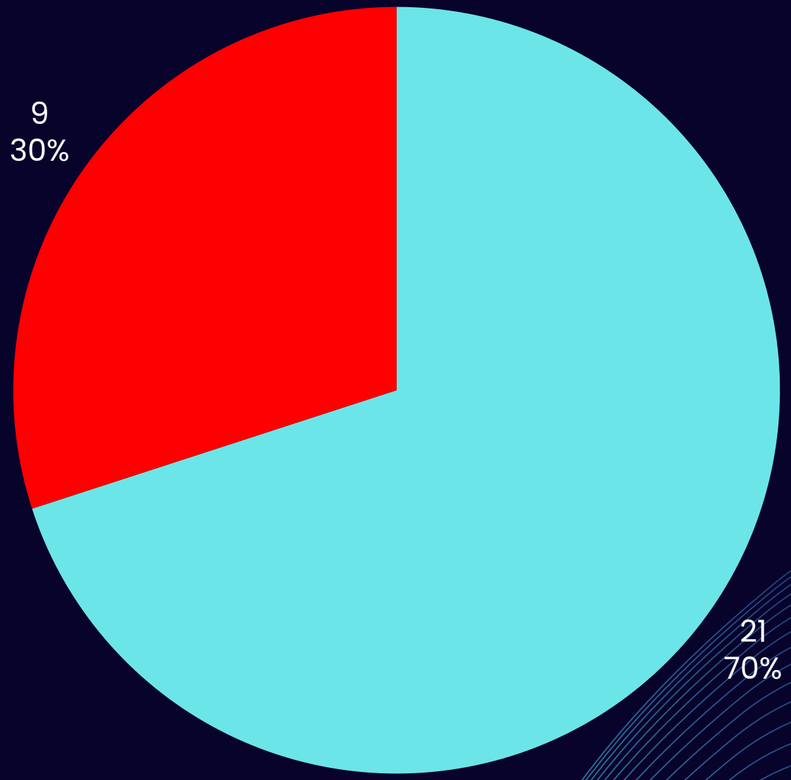
- 30 participants from diverse backgrounds (gender, major, year)

02 Tasks/Trials

- Analytical (3GB .csv File)
- Reasoning (Warehousing Problem)

03 Constraints

- Typing in absence of direct copy-pasting
- Time (15 min timer)



Experimental Design

03 Tools

- Python-based coding environment
- Access to a Large Language Model such as GPT-4.0
- Data was provided
- Raspberry Pi mounted eye-tracker
- Activity Tracker to detect keystrokes

04 Data Logging

- Eye tracking data includes pupil dilation as well as franticness of gaze.
- Prompts, responses, errors, and time to completion will be recorded
- Screen switches, backspaces, and number of code runs will be recorded
- Post experimental study NASA-TLX survey to validate the findings

Feedback from Pilot Study (n = 5)

ANALYTICAL

Analysis of Transaction data

- Break down larger tasks into related sub-tasks
- Ask for different kinds of visualizations
- Identify specific entries with the given constraints

REASONING

Warehouse Inventory Management

- Query in regards to a priority order
- Long-term planning
- Context specific categorizations

Tasks Details

ANALYTICAL

Analysis of Transaction data

- Statistical Analysis - Mean, median, standard deviation and counts
- Visualization of Monthly Transactions by Type
- Identifying specific Client ID with constrains
- Identifying specific Transaction ID with constraints

REASONING

Warehouse Inventory Management

- Identify Items with Critical Stock Levels
- Calculate Restocking Needs with Minimum Batches
- Categorize Items Based on Restocking Priority
- Generate a Consolidated Shipping Plan

Data & Features

Pupillometry

1

Data Collection

- Raspberry PI Infrared IR Night Vision Surveillance Camera Module 500W Webcam
- DeepLab v3+ Model

2

Preprocessing

- Noise Removal (< 200 px^2)
- Smoothening using Savitzky-Golay
- Normalization to [0 - 1]

3

Stress Analysis

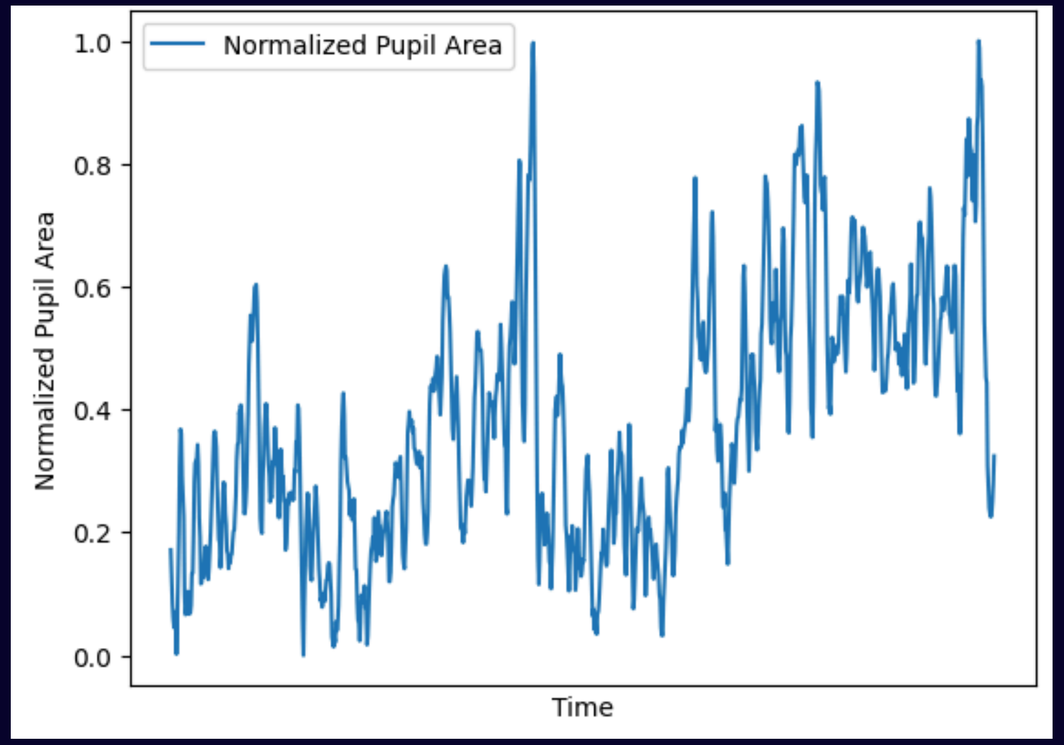
- Key Feature Extraction (mean of the first 15 seconds, std, max)
- Stress Thresholding

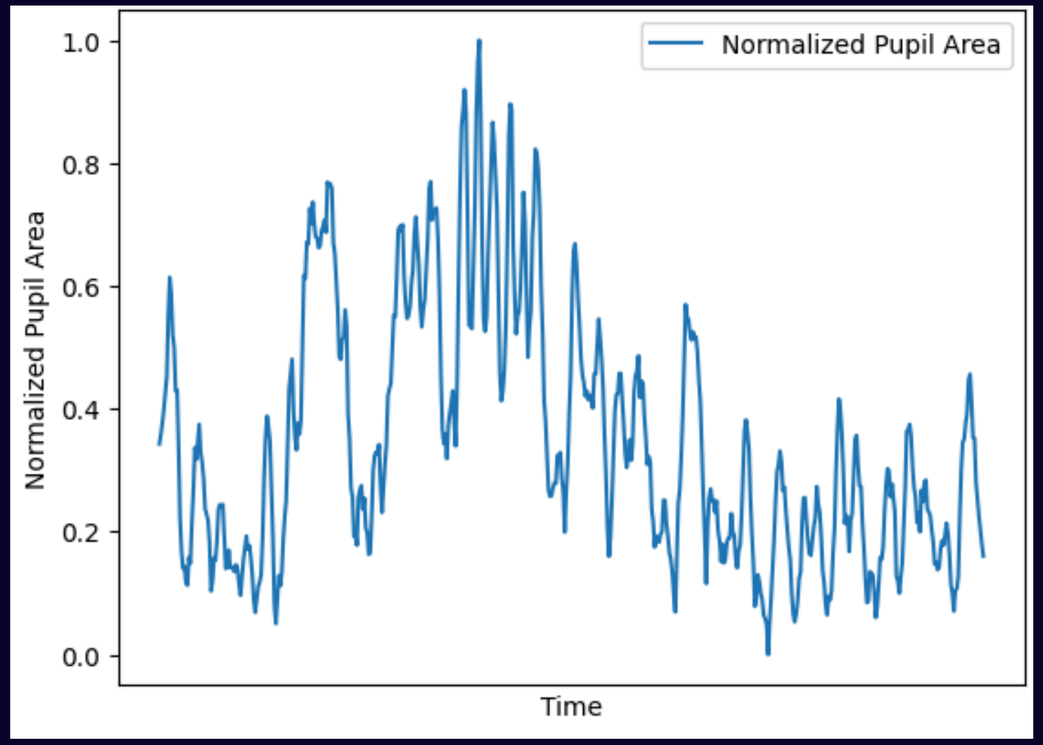
4

“Franticness”

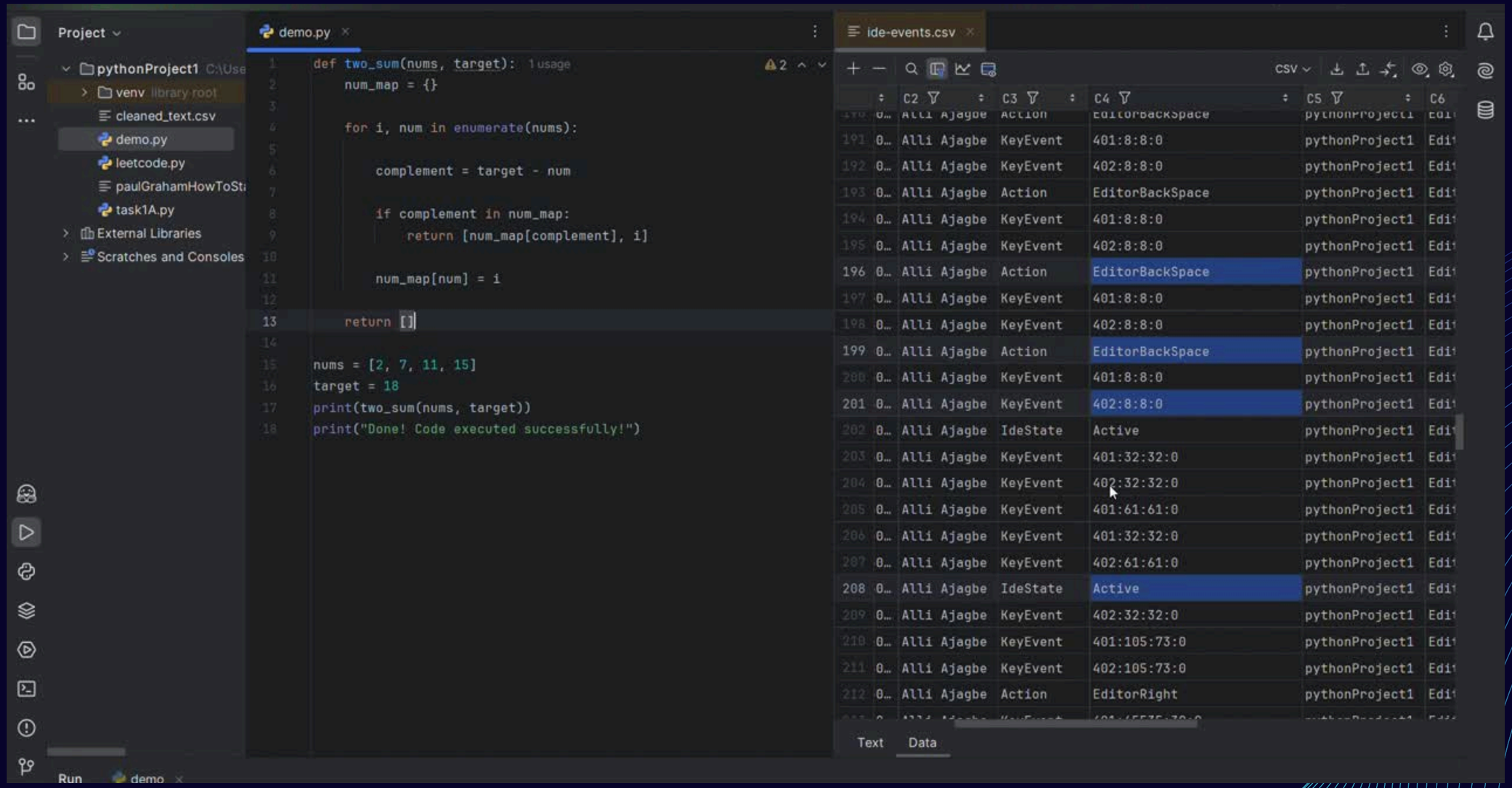
- Euclidean Distance in Consecutive px and py values
- Bucketing into Low, Medium, & High

Pupillometry





Behavioural Data Collection from IDE



ActivityTracker

01 Keyboard tracking
- Backspace Frequency - Error Rate
- Keywords Used

02 Window Switching
- Frequency of switching between ChatGPT and IDE

03 Mouse Tracking
- Number of Executions
- Frequency of Scrolls

04 Completion Time
- Task Start Time
- Task End Time

05 Code Length
- Number of Rows
- Number of Columns

06 Code Execution Time
- Runtime

Prompting Behaviour

Prompt Characteristics

01 Number of Prompts

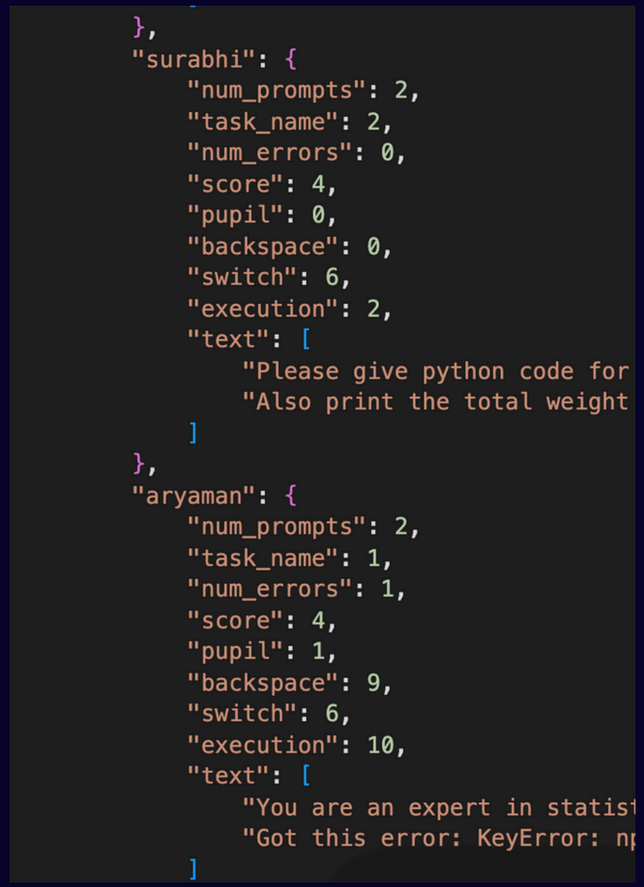
02 Length of Prompts - in terms of words

03 Number of errors

04 Usage of “I”, “You” and other keywords

05 Number of verbs

06 Task Score/Accuracy



Created Json with prompt features

Sample Prompts

“You”

You are an expert in statistics, and visualisation, who has been doing this for the past 15 years. I will be giiving you the dataset (columns) in context, and i need to you give me detailed end to end code for the following questions, that I require for my assignment:

“I”

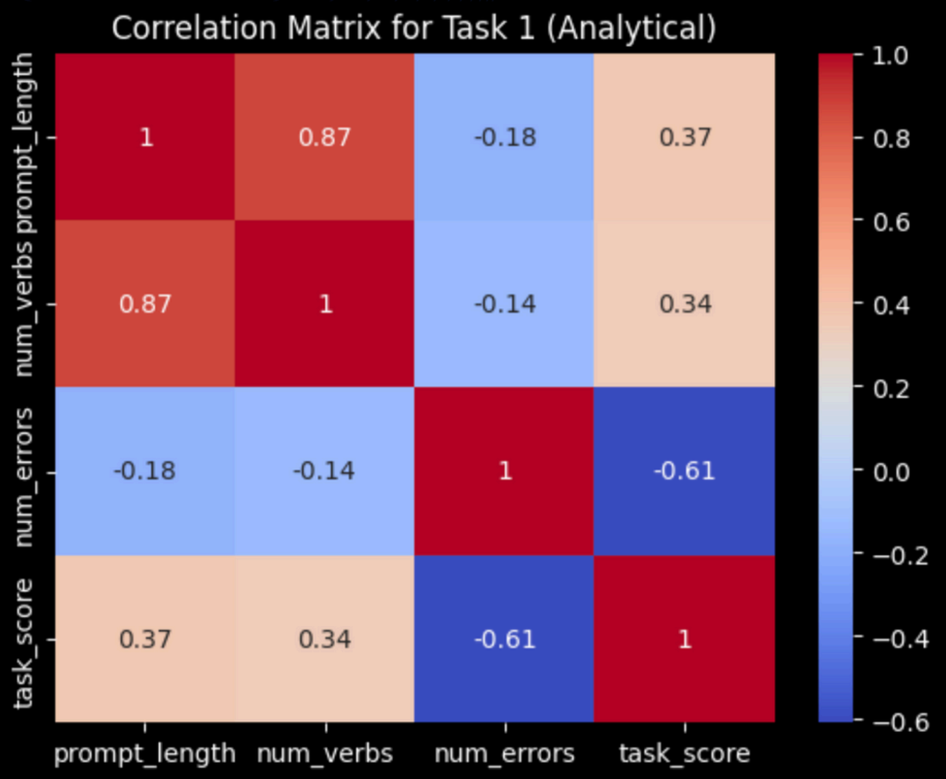
I have to identify all the items that are below a specified threhold. The threshold varies for different categories as fruits: 15, vegetables: 20 and grains: 10.

“please”

give me code for all these tasks please thanks

Key Question: Which prompting features correlate with high task accuracy and low cognitive load?

Prompting features, task score and errors



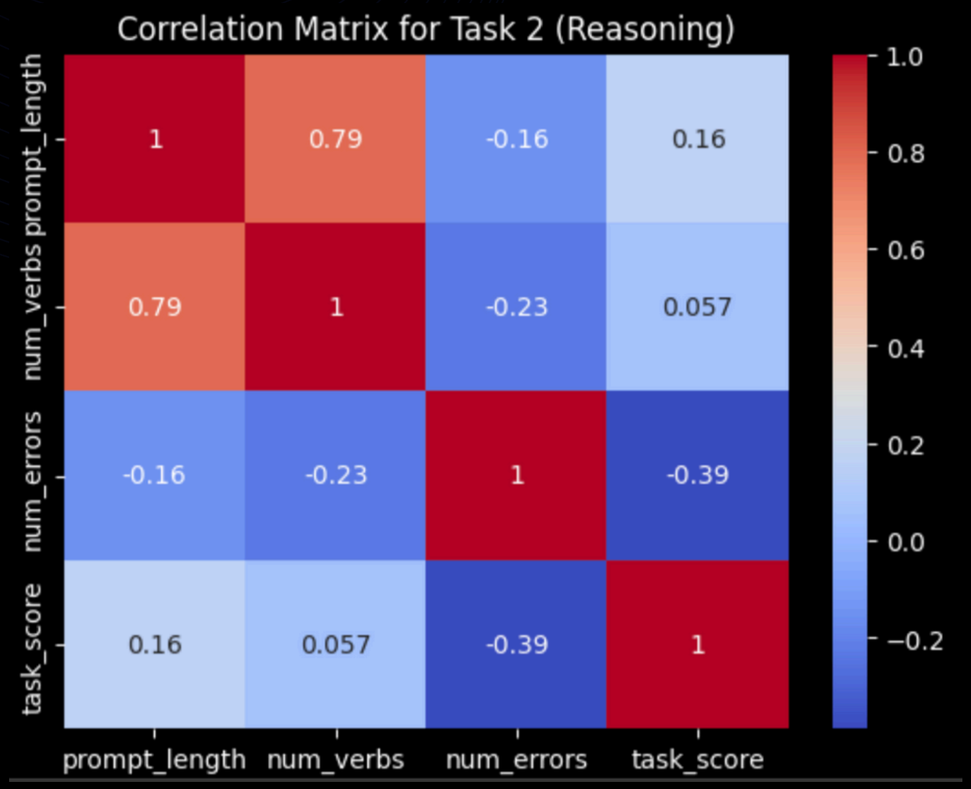
- In analytical tasks, longer prompts tend to perform better
- Detailed prompts

A longer, well-structured prompt helps ChatGPT understand complex instructions

- Errors significantly reduce the task score
- Implies that ChatGPT fails at fixing errors to give the correct output

Analytical

Prompting features, task score and errors



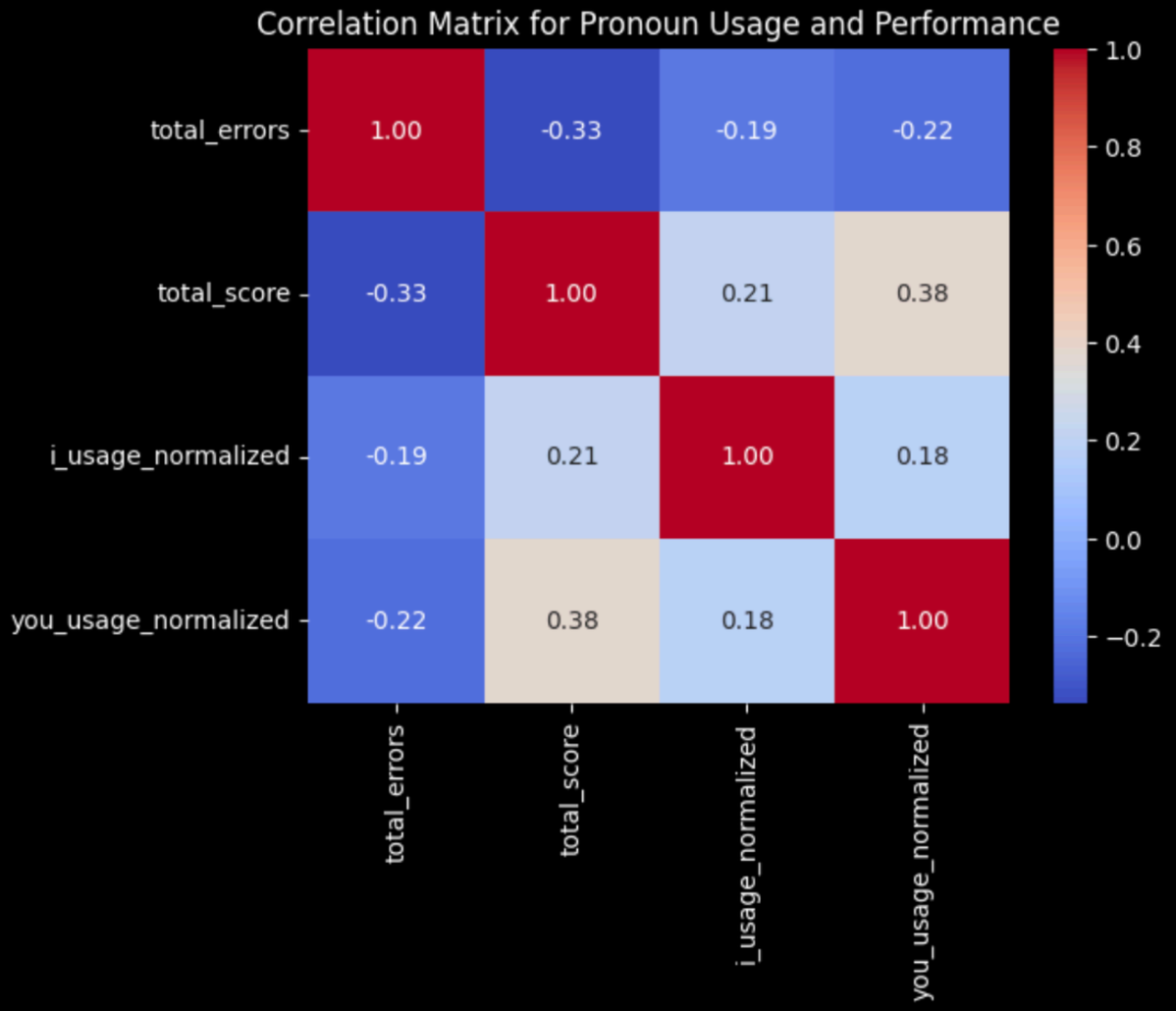
Reasoning

- Unlike analytical tasks, longer prompts do not strongly impact task scores in reasoning tasks

  A short, precise prompt may perform as well as a long one

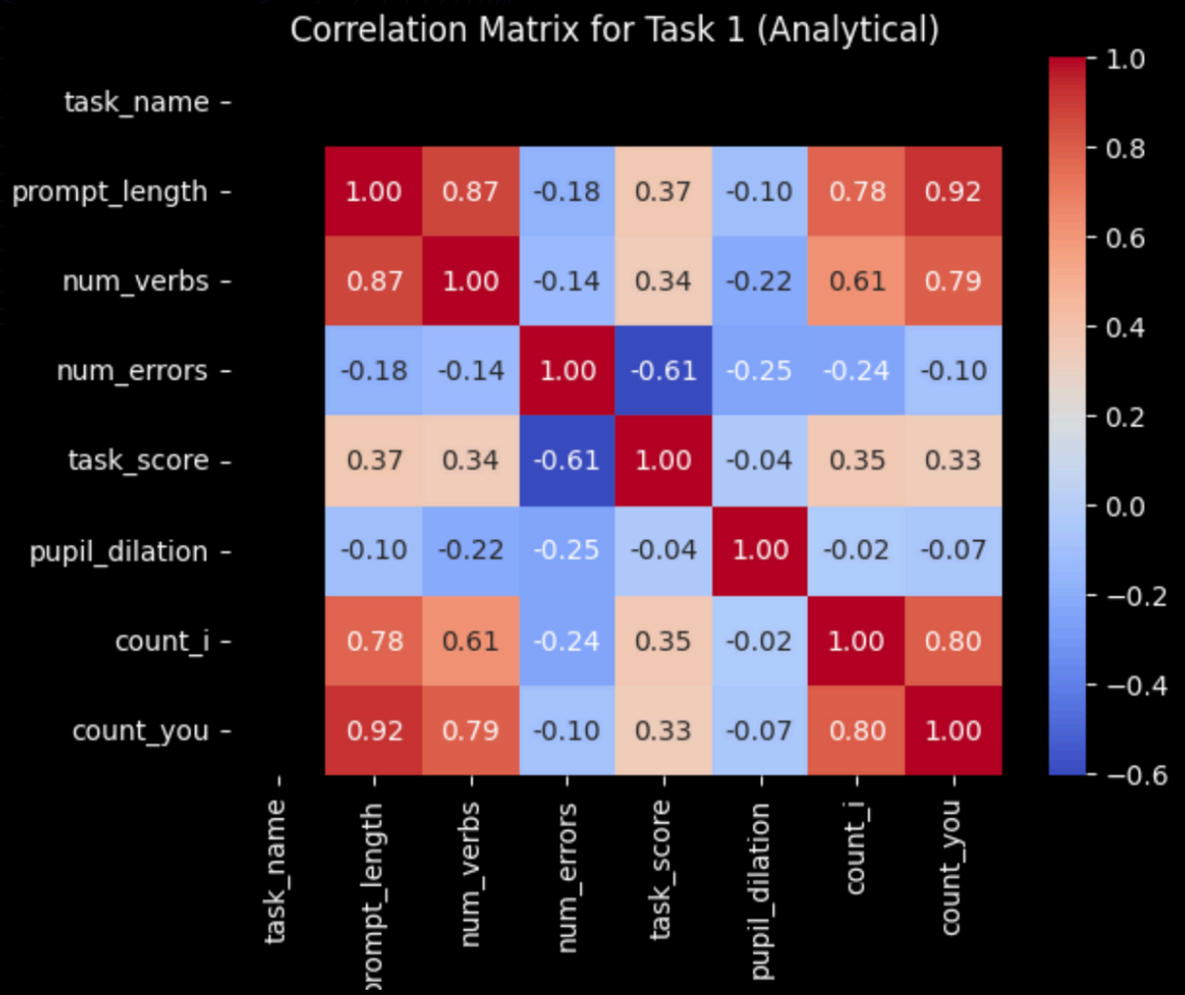
- Errors still reduce task scores, but the impact is less severe than in analytical tasks.

“I” usage vs “you” usage



- "You" Usage: Addressing ChatGPT directly seems to provide clearer instructions, improving task outcomes
- "I" Usage Has a Weaker Positive Correlation
- Higher "you" usage is linked to fewer errors, possibly because prompts are more instructional and precise.

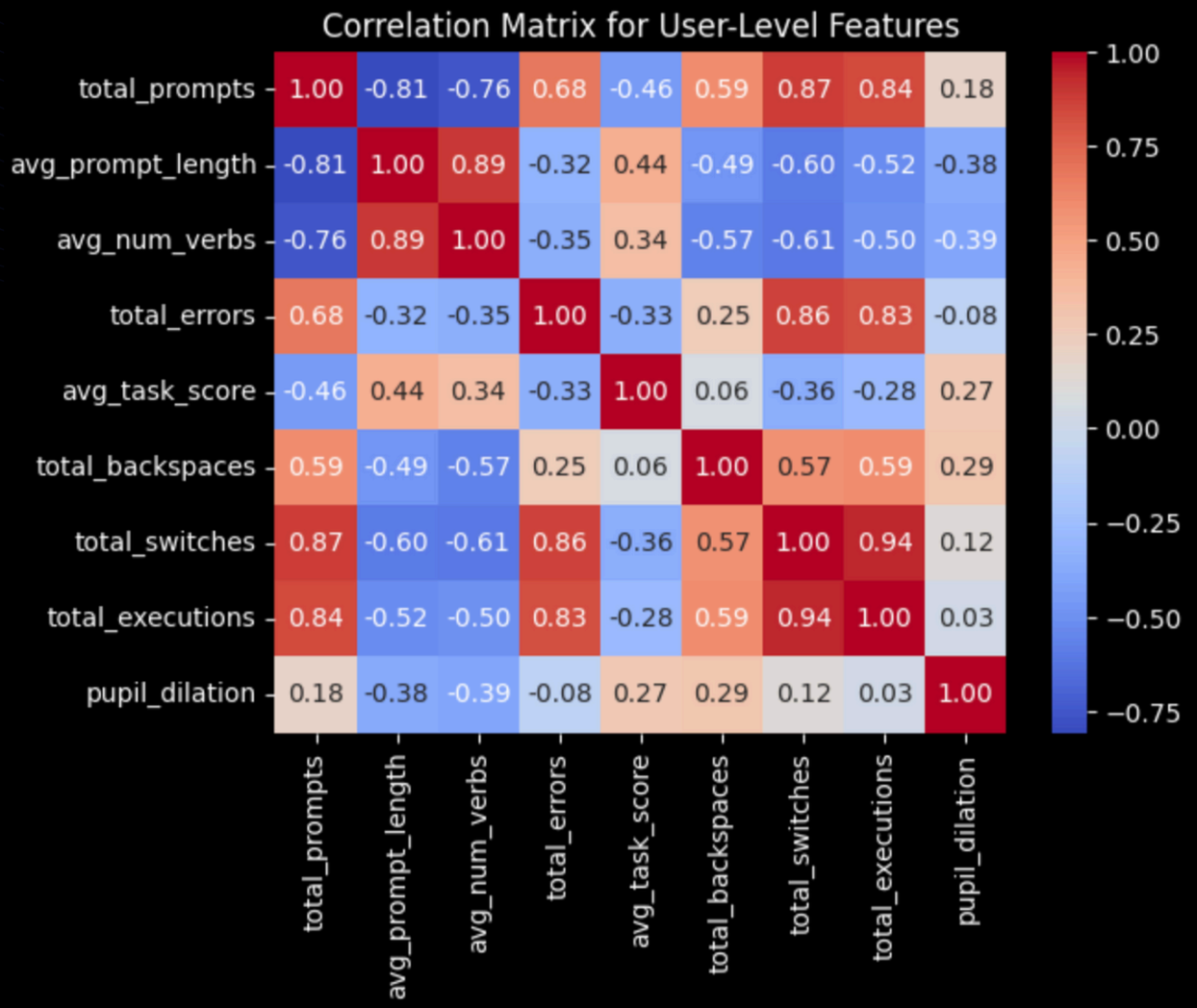
Relationships with Pupil dilation



- The number of verbs is negatively correlated with pupil dilation

Using more verbs while prompting may aid in reduced cognitive load

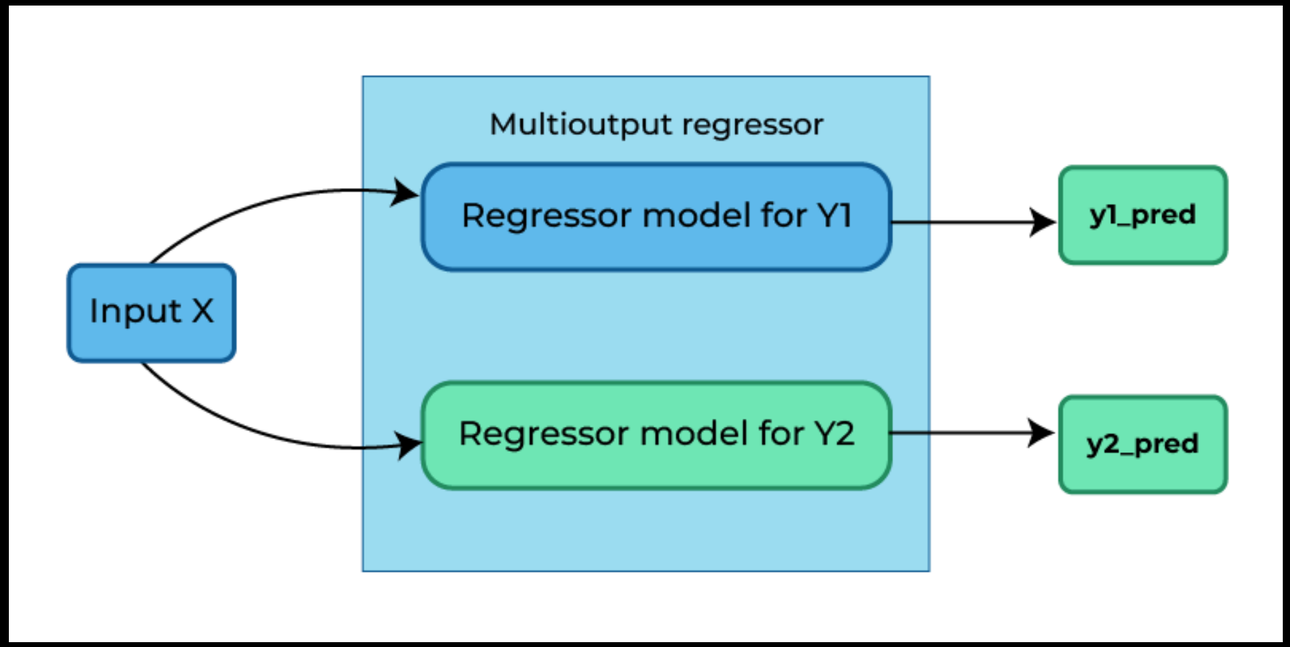
Relationships with Behavioural Characteristics



- Frequent backspaces are associated with higher cognitive load (stress).
- More screen-switches are strongly associated with higher errors.
- Frequent switches reduce task accuracy.

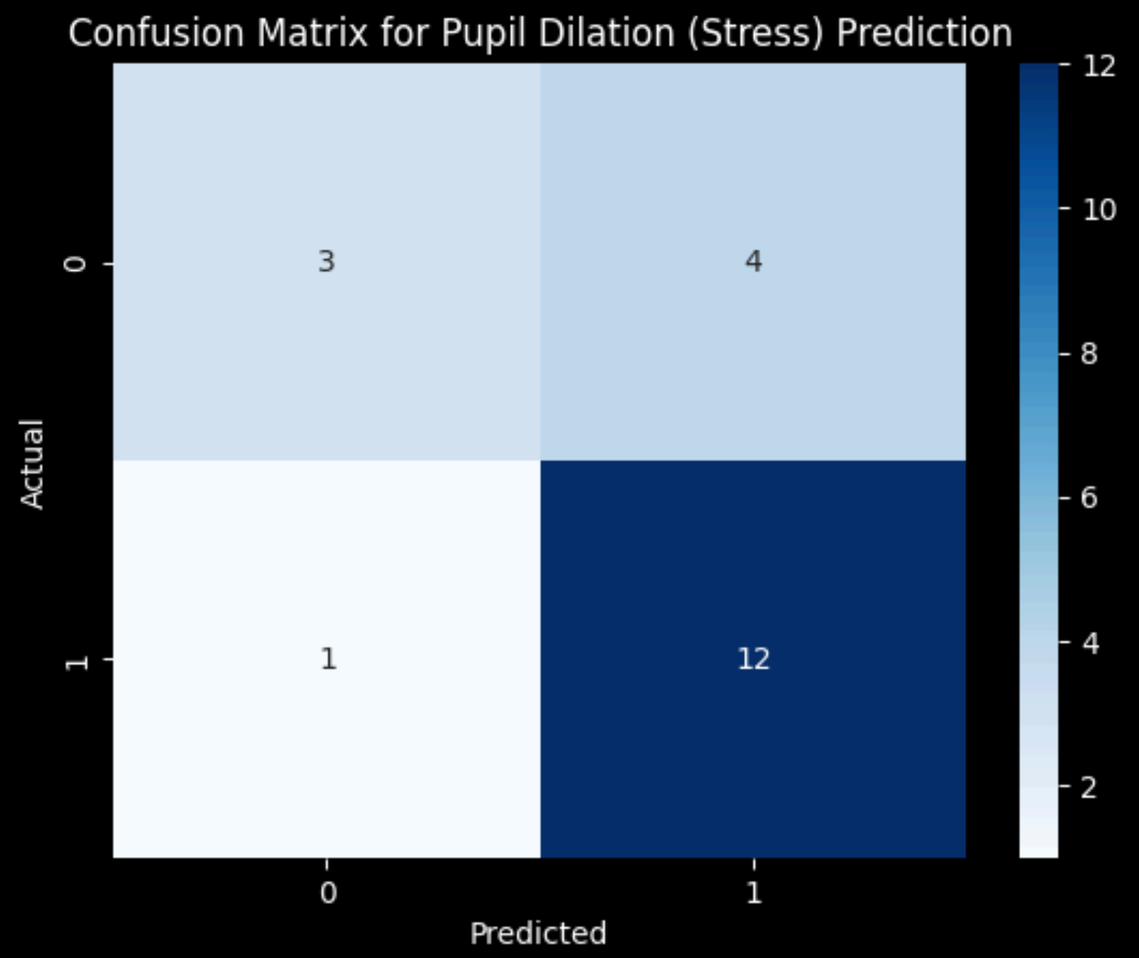
Multi-output Regression

- Mixed Outputs
- Interdependency: Cognitive load can directly impact task accuracy, making joint modeling meaningful.



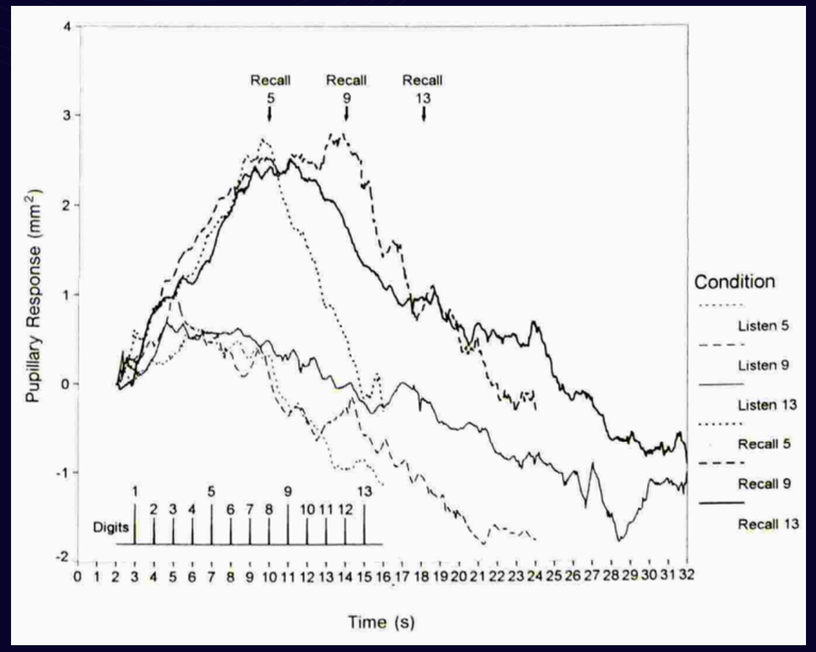
Multi-output Regression

Metrics:
Task Accuracy - MAE: 0.1220, RMSE: 0.3493
Pupil Dilation - Accuracy: 75%

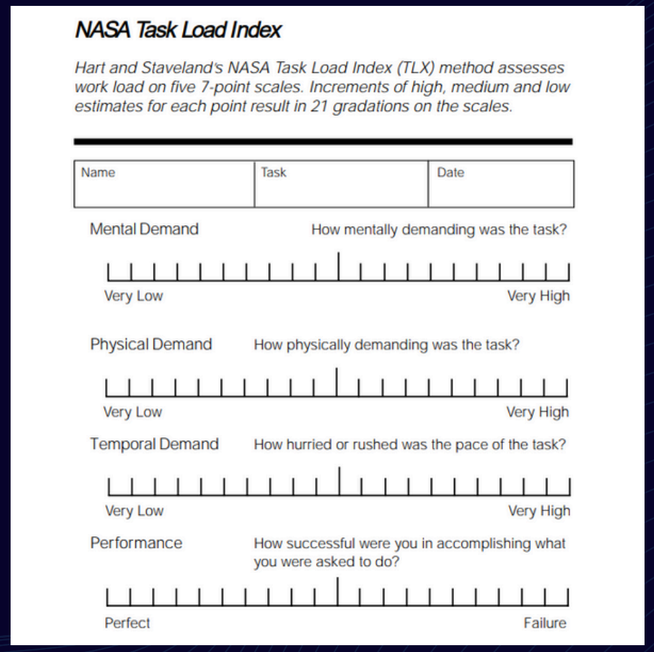


Target and Validation

"Task-invoked pupillary response is one such physiological response of cognitive load on working memory, with studies finding that pupil dilation occurs with high cognitive load."



For all digit spans, the pupillary response increases as the subjects listen to the digits, indicating an increase in cognitive load.



It is a subjective self-reporting set of scores that will be used to validate the measure of cognitive load from pupil dilation

GRANHOLM, E., ASARNOW, R. F., SARKIN, A. J., & DYKES, K. L. (1996). *Pupillary responses index cognitive resource limitations.*

Challenges

- Data Collection
  - Configuration Issues with Hardware
  - Synchronising Pupil Data and IDE tracking data
  - Participants Familiarity with Mac Laptop
- Bias in NASA TLX survey
- Sample Size

Deployment – A Close Example

Potential Impacts

- **Direct impact on Cognitive Load Management**
  - Improved Understanding of LLM Interaction
  - Enhanced Developer Productivity

- The experimental analysis shows that **longer, well-structured prompts** improve performance in analytical tasks while **shorter, concise prompts** suffice for reasoning tasks.

By optimizing LLM interactions, developers can focus on problem-solving and innovation instead of struggling with LLM-induced challenges, fostering faster prototyping and experimentation. Developers can now tailor their prompt design based on task complexity, leading to improved LLM output accuracy and efficiency.

Thank you!