BACHELOR PAPER

Thesis submitted in fulfillment of the requirements for the degree of Bachelor of Science in Engineering at the University of Applied Sciences Technikum Wien Degree Program Computer Science Dual

Comparing Chain-of-Thought, Chain-of-Verification and Self-Refine in solving Brazilian University Entrance Exams

By: Rafaela Rolim Santana

Student Number: 2210257114

Supervisor: Patrick Link, BSc.

Vienna, 02.04.2025

Declaration of Authenticity

“As author and creator of this work to hand, I confirm with my signature knowledge of the relevant copyright regulations governed by higher education acts (see Urheberrechtsgesetz/ Austrian copyright law as amended as well as the Statute on Studies Act Provisions / Examination Regulations of the UAS Technikum Wien as amended).

I hereby declare that I completed the present work independently and according to the rules currently applicable at the UAS Technikum Wien and that any ideas, whether written by others or by myself, have been fully sourced and referenced. I am aware of any consequences I may face on the part of the degree program director if there should be evidence of missing autonomy and independence or evidence of any intent to fraudulently achieve a pass mark for this work (see Statute on Studies Act Provisions / Examination Regulations of the UAS Technikum Wien as amended).

I further declare that up to this date I have not published the work to hand nor have I presented it to another examination board in the same or similar form. I affirm that the version submitted matches the version in the upload tool.”

|  |  |  |
| --- | --- | --- |
|  |  |  |
| Place, Date |  | Digital Signature |

Abstract

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Maecenas porttitor congue massa. Fusce posuere, magna sed pulvinar ultricies, purus lectus malesuada libero, sit amet commodo magna eros quis urna.

Nunc viverra imperdiet enim. Fusce est. Vivamus a tellus.

Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Proin pharetra nonummy pede. Mauris et orci.

Aenean nec lorem. In porttitor. Donec laoreet nonummy augue.

Suspendisse dui purus, scelerisque at, vulputate vitae, pretium mattis, nunc. Mauris eget neque at sem venenatis eleifend. Ut nonummy.

Fusce aliquet pede non pede. Suspendisse dapibus lorem pellentesque magna. Integer nulla.

Donec blandit feugiat ligula. Donec hendrerit, felis et imperdiet euismod, purus ipsum pretium metus, in lacinia nulla nisl eget sapien. Donec ut est in lectus consequat consequat.

Acknowledgements

**Keywords:** Keyword1, Keyword2, Keyword3, Keyword4, Keyword5

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Maecenas porttitor congue massa. Fusce posuere, magna sed pulvinar ultricies, purus lectus malesuada libero, sit amet commodo magna eros quis urna.

Nunc viverra imperdiet enim. Fusce est. Vivamus a tellus.

Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Proin pharetra nonummy pede. Mauris et orci.

Aenean nec lorem. In porttitor. Donec laoreet nonummy augue.

Table of Contents

[1 This is the heading of the first chapter **Error! Bookmark not defined.**](#_Toc179533587)

[1.1 Heading level 2 **Error! Bookmark not defined.**](#_Toc179533588)

[1.1.1 Heading level 3 **Error! Bookmark not defined.**](#_Toc179533589)

[1.1.1.1 Heading level 4 **Error! Bookmark not defined.**](#_Toc179533590)

[2 This is the heading of the second chapter **Error! Bookmark not defined.**](#_Toc179533591)

[2.1 Heading level 2 **Error! Bookmark not defined.**](#_Toc179533592)

[2.1.1 Heading level 3 **Error! Bookmark not defined.**](#_Toc179533593)

[2.1.1.1 Heading level 4 **Error! Bookmark not defined.**](#_Toc179533594)

[Bibliography **Error! Bookmark not defined.**](#_Toc179533595)

[List of Figures 14](#_Toc179533596)

[List of Tables 15](#_Toc179533597)

[List of Abbreviations 16](#_Toc179533598)

[Documentation table of AI-based tools 17](#_Toc179533599)

[A: Heading of Appendix A 18](#_Toc179533600)

[B: Heading of Appendix B 19](#_Toc179533601)

# Introduction

In recent years, Large Language Models (LLMs) such as GPT-4 have demonstrated impressive capabilities across various tasks, including natural language understanding, code generation, and interactive dialogue systems. Despite these successes, a significant challenge remains: LLMs frequently produce plausible but inaccurate or misleading outputs, a phenomenon known as hallucination. Hallucinations can severely impact the applicability and trustworthiness of LLM-generated information, particularly in educational contexts. Students increasingly rely on LLM-generated answers for learning and assessment purposes; thus, inaccuracies can mislead their understanding, reinforce misconceptions, and ultimately hinder their academic progress.

To address these limitations, recent literature has explored innovative prompting strategies aimed at enhancing the reasoning and verification capabilities of LLMs. Notable methods include Chain-of-Thought (CoT), which encourages models to generate step-by-step reasoning sequences [1]; Chain-of-Verification (CoVe), where models explicitly verify intermediate reasoning steps [2]; and Self-Refine, which iteratively improves model outputs using self-generated feedback [3]. These methods show potential in systematically improving the reliability and accuracy of LLM-generated responses.

This thesis aims to conduct a comparative analysis of these three prompting methods, evaluating their effectiveness in improving answer accuracy in multiple-choice question-answering tasks. Specifically, the research seeks to answer the following questions:

 **RQ1**: How does the application of Chain-of-Thought, Chain-of-Verification and Self-Refine prompting techniques affect LLM response accuracy on Brazilian University Entrance Exams?

 **RQ2**: How does the effectiveness of Chain-of-Thought, Chain-of-Verification, and Self-Refine prompting techniques vary across different subject areas?

To achieve these goals, this thesis will adopt an experimental approach using standardized QA datasets. Experiments will compare quantitative measures such as general accuracy and accuracy by subject, and statistical analyses are performed to determine if significant performance differences exist between these techniques.

# Methodology

## Ideal Solution

A rigorous evaluation of prompting techniques for Large Language Models (LLMs) requires a controlled and repeatable experimental setup that allows fair, transparent, and statistically valid comparisons across methods. In an ideal scenario, the following conditions would be met:

* **Fully deterministic LLM behavior:** Each prompt would yield a predictable and reproducible response, eliminating randomness and isolating the effect of prompt structure alone.
* **Access to model internals:** Evaluators would have control over the underlying LLM architecture, token-level probabilities, and intermediate reasoning states, allowing for in-depth analysis.
* **Large-scale, domain-specific benchmark datasets:** Evaluation would be conducted on diverse, high-quality questions across multiple disciplines, ideally with rich annotations such as reasoning steps and difficulty levels.
* **Automated multi-pass evaluation:** The system would support large-scale, automated prompting with batch processing, minimal API latency, and robust error handling.
* **Statistical power and interpretability:** Sufficiently large numbers of prompt-response pairs would be generated per method to ensure statistical confidence, and the analysis would include meaningful performance metrics beyond accuracy, such as consistency and reasoning quality.

In practice, evaluating LLM behavior under these ideal conditions is constrained by several factors, including API access limitations, model stochasticity, and computational cost. Therefore, this study adopts a **practical approximation** of the ideal solution by combining well-defined prompting templates, a standardized evaluation dataset, and repeated test runs. Multiple generations per question were used to estimate model consistency, and statistical tests were employed to validate significance across methods and subject areas.

The goal of this setup is not to eliminate uncertainty entirely, but to **systematically control for variability** while enabling reproducibility and transparent comparison across Chain-of-Thought, Chain-of-Verification, and Self-Refine prompting strategies.

## Requirements

explain how i did a literature review to select the techniques

how these helped me choose the dataset, design metrics and prompting templates

## Development Process

This section describes the practical steps taken to implement the evaluation framework, from preparing the dataset to executing the prompts and collecting results.

### Dataset Preparation

The experiments were conducted using the publicly available ENEM 2024 dataset [4], which contains all multiple-choice questions from the Brazilian national university entrance exam (Exame Nacional do Ensino Médio). A total of **180 questions** were used, covering four official subject areas:

* Linguagens, Códigos e suas Tecnologias (Languages)
* Ciências Humanas e suas Tecnologias (Human Sciences)
* Ciências da Natureza e suas Tecnologias (Natural Sciences)
* Matemática e suas Tecnologias (Mathematics)

Each question included the question, the multiple choice alternatives and a ground-truth label. The dataset was loaded from a .jsonl file and parsed using Python. Subject labels were assigned programmatically based on question order, reflecting the fixed structure of ENEM exams.

### Prompt Template Definition

To ensure consistency and control over input formatting, **custom structured prompting templates** were created for each of the three methods under evaluation:

* **Chain-of-Thought (CoT):** Used a 3-shot prompt with explicit step-by-step reasoning.
* **Chain-of-Verification (CoVe):** Extended the CoT format by generating and answering follow-up verification questions before revising them for the final answer.
* **Self-Refine:** Combined CoT-style reasoning with an iterative feedback–refinement loop, where the model reviewed and improved its own answers.

Each template was implemented using Python functions that dynamically populated prompt slots with ENEM question data. These templates formed the backbone of the evaluation.

### Testing Phase

Prompts were executed using the **OpenAI API**, targeting GPT-3.5 as the underlying model. Each question was run **multiple times (38 per method)** to capture variability and assess consistency. Each response was recorded along with relevant metadata such as timestamps, reasoning traces, and intermediate answers.

The execution pipeline was implemented in **Jupyter Notebooks**, and run-specific results were saved to timestamped CSV files for downstream processing.

### Evaluation and Data Collection

For each prompt-response pair, the following data points were collected:

* The full model output (including reasoning and answer),
* The extracted answer letter (A–E),
* Whether the prediction was correct (compared to the ground-truth),
* The subject label (for subject-specific analysis),
* Additional metadata depending on method:
  + **CoVe:** Verification questions and answers
  + **Self-Refine:** Iterative responses, feedback trace and intermediate answers
  + **CoT:** Raw CoT response

This structured dataset enabled quantitative evaluation across multiple dimensions, including overall accuracy, subject-wise performance, and prediction consistency. All further analyses and visualizations were performed using **pandas, matplotlib**, seaborn, numpy, and **scipy**.

## Tests

The evaluation framework was designed to measure the effectiveness and reliability of each prompting method using three core metrics: overall accuracy, subject-specific accuracy, and consistency across repeated runs. Each metric targets a different dimension of performance.

### General Accuracy

General accuracy was calculated as the proportion of correct answers produced by each method across all ENEM questions and all runs. For each method, accuracy was computed per run (based on 180 questions), and then averaged across the 38 runs to obtain the mean accuracy and standard deviation. This approach captures both central performance and run-level variability, enabling fair comparison across methods.

This metric directly addresses **Research Question 1 (RQ1)**, which investigates whether certain prompting techniques lead to more accurate predictions overall. By aggregating performance across the full test set, general accuracy provides a high-level indicator of each method's effectiveness.

### Accuracy by Subject

To assess how performance varied by academic domain, accuracy was also computed separately for each of the four official ENEM subjects. For every method and run, subject-specific accuracy was calculated as the mean correctness over questions in that subject area. These per-run values were then aggregated to compute the mean and standard deviation of accuracy per subject and method.

This subject-wise breakdown was essential to answer **Research Question 2 (RQ2)**, which investigates whether prompting strategies differ in effectiveness depending on the domain of the question.

### Consistency

Consistency measures how reliably a prompting method produces the **same answer** to the **same question** across different runs. For each method, a question was marked as consistent if the same answer was predicted in all 38 runs. The overall consistency rate was computed as the proportion of questions classified as consistent.

To further investigate domain-specific stability, consistency was also computed **per subject** by grouping questions by their subject label and calculating the proportion of consistently answered items within each group.

This metric was critical to answering **Research Question 3 (RQ3)**, which addresses the **reliability** of each prompting method under repeated evaluation.

## Tools

All experiments, data processing, and analyses were conducted using openly available tools and Python libraries. The following technologies were used throughout the development and evaluation process:

* **Jupyter Notebook**  
  Provided an interactive environment for developing, testing, and documenting the experimental pipeline. It facilitated reproducibility and allowed step-by-step analysis of intermediate outputs, errors, and statistical results.
* **OpenAI API**  
  Used to generate model responses via the GPT-3.5 language model. The API enabled programmatic submission of prompts and retrieval of outputs across multiple runs and prompting strategies.
* **pandas**  
  Served as the primary tool for data manipulation, grouping, filtering, and cleaning of results. It was used extensively to structure run outputs, calculate performance metrics, and organize data for visualization.
* **matplotlib** and **seaborn**  
  These libraries were used for generating all plots and visualizations in the analysis. Bar plots, box plots, and significance markers were created to communicate results in a clear and informative way.
* **numpy**  
  Provided support for efficient numerical operations, including array-based computations and statistical aggregation.
* **scipy**  
  Used for statistical testing, including Shapiro–Wilk tests for normality, Levene’s test for variance homogeneity, Kruskal–Wallis tests for group comparison, and McNemar’s tests for paired binary outcomes.

# Solution

## Answering the Research Questions

This study investigates the impact and effectiveness of different prompting techniques when applied to large language models (LLMs) solving high-school level multiple-choice questions from the 2024 Brazilian university entrance exam (ENEM). Three state-of-the-art prompting strategies were selected and systematically compared: Chain-of-Thought (CoT), Chain-of-Verification (CoVe), and Self-Refine. The research is guided by the following three questions:

**RQ1:** *How does the application of CoT, CoVe and Self-Refine affect LLM response accuracy on Brazilian University Entrance Exams?*

**RQ2:** *How does the effectiveness of CoT, CoVe and Self-Refine vary across different subject areas?*

**RQ3:** *How consistent are the predictions of Chain-of-Thought, Chain-of-Verification, and Self-Refine methods across repeated runs?*

Each method was implemented as a reusable prompt template and applied to the same standardized question set across multiple runs. The evaluation strategy is based on three core criteria:

**Accuracy**: The proportion of correct responses produced by each method across all questions.

**Subject-specific Performance**: The accuracy and consistency of each method when broken down by academic subject (Mathematics, Human Sciences, Languages, and Natural Sciences).

**Consistency**: The ability of a method to produce the same answer for the same question across repeated test runs.

By applying each prompting technique to the same ENEM question set and evaluating them through these complementary metrics, the study enables a comparison of both overall effectiveness and domain-specific behavior. This approach ensures that not only are general performance trends captured, but also method-specific strengths or weaknesses in particular subject areas.

Together, these analyses provide a comprehensive answer to RQ1 and RQ2. They help identify which technique produces the most accurate, stable, and subject-adaptable results, thereby highlighting the most promising strategy for enhancing reliability in LLM-assisted educational tasks.

## Prompting Techniques

The implementation of prompting techniques involved defining templates for each of the three methods: Chain-of-Thought (CoT), Chain-of-Verification (CoVe), and Self-Refine. Each template was designed to guide the LLM to generate responses in specific structured formats to facilitate comparative analyses.

### ****Chain-of-Thought (CoT)****

**Definition and Rationale**

Chain-of-Thought (CoT) is a prompting technique designed to improve the reasoning capabilities of large language models (LLMs) by encouraging them to produce step-by-step explanations before arriving at a final answer. Rather than directly outputting a single response, CoT prompts guide the model through intermediate reasoning steps that mimic human problem-solving behavior. According to *Chain-of-Thought Prompting Elicits Reasoning in Large Language Models* [1] , this structured form of output has shown to significantly improve performance on tasks requiring logical inference, arithmetic, and multi-step analysis.

**Key Steps in the Process**

In the CoT approach, the model follows a three-stage process:

1. **Instructional Framing**:

The model is introduced to the task through a few-shot prompt containing example ENEM questions and detailed explanations.

1. **Reasoned Answer Generation**:

The model is expected to produce a chain of reasoning that explains the thought process leading to its answer.

1. **Explicit Final Answer**:

After the reasoning steps, the model is prompted to select and declare the correct multiple-choice alternative.

**Implementation for ENEM Questions**

In this study, CoT was implemented using a reusable Python prompt template tailored for ENEM-style multiple-choice questions. Each prompt included three few-shot examples taken directly from the work *Evaluating GPT-3.5 and GPT-4 Models on Brazilian University Admission Exams* [4], where the authors evaluated GPT-3.5 and GPT-4 models on the ENEM using structured prompts. These examples were carefully selected from the **ENEM 2022 exam** and span across three major subject domains: **Languages, Human Sciences,** and **Mathematics**. Each example is formatted to reflect the CoT structure: the question is followed by the answer alternatives, a detailed explanation that analyzes each option, and a final line that explicitly states the correct alternative in a structure format.

To maintain the representativeness and reliability of these demonstrations, the explanations were derived from expert teacher discussions and public exam commentary resources, as also documented in the paper [4]. In the prompt template, the few-shot block is followed by the question under evaluation, inserted dynamically during each test run using the build\_cot\_prompt() function. The model is instructed to produce a full reasoning chain and clearly identify the final answer at the end.

This template was used across **multiple iterations**, allowing the same questions to be answered by the model in different runs to evaluate consistency. Each response was stored with metadata including the selected alternative and the full generated reasoning chain for further analysis of both correctness and coherence.

**Representativeness of the Format**

The CoT template used in this study is representative of the method’s core design principles. It follows the formulation proposed by Wei et al. [1] by including few-shot reasoning examples and encouraging natural language explanations prior to final answer selection. Furthermore, by using real-world educational content from the ENEM, the template ensures that the reasoning steps required are diverse, realistic, and context-rich—qualities that highlight the strengths and limitations of CoT in practical use cases. This makes the experimental setup well-suited for evaluating CoT’s robustness and generalizability across different academic subjects.

### ****Chain-of-Verification (CoVe)****

**Definition and Rationale**

Chain-of-Verification (CoVe) is a prompting method developed to reduce hallucinations in large language models (LLMs) by having the model **deliberate over its own outputs**. Introduced in *Chain-of-Verification Reduces Hallucination in Large Language Models* [2], CoVe operates on the assumption that LLMs can critically assess and improve their responses when prompted appropriately. Rather than relying on a single forward pass, CoVe decomposes the generation process into deliberate verification steps. This makes the method particularly promising in educational or factual domains where high answer reliability is crucial.

**Key Steps in the Process**

The CoVe process involves **four sequential stages** [2]:

1. **Generate a Baseline Response:**  
   An initial draft is generated using a standard prompting technique (in this study, 3-shot Chain-of-Thought).
2. **Plan Verification Questions:**  
   The model reflects on the baseline response and produces verification questions aimed at checking factual components or reasoning steps.
3. **Execute Verifications:**  
   Each verification question is answered in isolation. This step is critical, as it prevents the model from being biased by its own previous answers.
4. **Generate Final Verified Response:**  
   Using the original question, the baseline response, and the results of the verifications, the model revises and outputs a final, self-corrected answer.

**Implementation for ENEM Questions**

In this study, CoVe was implemented as a three-part Python pipeline operating on questions from the ENEM dataset. All steps are executed inside an iterative loop that repeats the CoVe pipeline over multiple test runs.

The process is initiated by generating a **baseline response** using the same few-shot Chain-of-Thought template described in the CoT section. This response serves as the initial candidate answer.

The second step uses the plan\_verification\_questions() function to generate targeted verification questions. The prompt instructs the LLM to reflect on its previous answer and identify key factual or logical claims worth verifying. These questions are generated without any fixed template, allowing the model to frame them in natural language.

Each verification question is then answered independently using the execute\_verifications() function. To ensure a robust verification process, each question is sent to the LLM in isolation, without providing the baseline response in the context—this emulates the "factored" verification setup, which was shown to prevent the model from repeating its hallucinations​ [2].

Finally, the generate\_final\_verified\_answer() function constructs a full prompt that includes the original question, alternatives, the baseline answer, and the verification Q&A pairs. The LLM is then asked to synthesize this information into a revised, fully verified response. This final step mimics the ***Factor + Revise*** variant, which the original authors found to yield the **highest factual accuracy** in longform settings [2].

**Representativeness of the Format**

This implementation adheres closely to the CoVe procedure described by *Chain-of-Verification Reduces Hallucination in Large Language Models* [2]. It separates the planning, execution, and revision stages into distinct prompting steps and avoids contamination between outputs, as recommended in the paper. Moreover, by adapting CoVe to a multiple-choice educational benchmark (ENEM), this study explores its effectiveness in a **structured, high-stakes, multilingual** domain—extending its applicability beyond the original open-domain QA and biography generation settings. Thus, the prompting format used here is not only faithful to the original CoVe proposal but also well-suited for evaluating its impact on **accuracy and consistency** in educational tasks.

### ****Self-Refine****

**Definition**

Self-Refine is an iterative prompting technique in which a language model improves its own output through cycles of self-critique and revision. Instead of producing a single-shot response, the model first generates an answer, then evaluates it by generating critical feedback, and finally uses that feedback to revise and refine its response. This method mimics how humans revise drafts and has been shown to improve LLM performance across diverse tasks, including reasoning, coding, and factual QA [3].

**Key Steps in the Process**

My implementation follows the core Self-Refine architecture, consisting of three repeated steps:

1. **Initial Generation**:

The model answers the ENEM question using the 3-shot Chain-of-Thought (CoT) template, producing an initial reasoning-based answer.

1. **Feedback Generation**:

The model receives a second prompt asking it to critically analyze its own response, pointing out factual mistakes, logical inconsistencies, or poor reasoning.

1. **Refinement**:

The model receives a third prompt to revise its answer, explicitly taking the generated feedback into account. It is instructed to re-express the reasoning and conclude with a final answer in the specified format.

This feedback–refinement loop is repeated until the selected multiple-choice answer remains unchanged between iterations, or until a maximum of 10 iterations is reached.

**Implementation for ENEM Questions**

The implementation was carried as a Python pipeline. The function run\_self\_refine() loops through all ENEM questions, applying the Self-Refine logic defined in self\_refine\_enem(). For each question, the process begins by calling build\_cot\_prompt() to create an initial 3-shot CoT response. Then, for up to 10 iterations:

* A feedback prompt is created using build\_feedback\_prompt(), asking the model to identify and critique weaknesses in its response.
* A refine prompt is created with build\_refine\_prompt(), instructing the model to revise its answer based on that feedback.
* If the final predicted answer remains the same as the one in the previous iteration, the loop stops.

Throughout this process, all iterations, responses, feedbacks, and selected answers are stored in a structured format, including metadata like subject and answer trace. This enables detailed post-analysis on consistency and progression.

**Why This Format is Representative**

This setup closely mirrors the original Self-Refine procedure [3], including the iterative feedback-refinement loop, stopping condition based on convergence, and use of in-context learning without fine-tuning. Applying this method to ENEM questions is a strong test of its effectiveness: the questions are diverse and challenging, often requiring contextual reasoning, logical structuring, and domain-specific knowledge — all aspects that benefit from iterative self-correction. Furthermore, including the full trace of intermediate answers allows for performance and consistency analysis aligned with the goals of this thesis.

## Load QA dataset

The question-answer dataset used in this study was sourced from the open-access Maritaca AI ENEM dataset [4], hosted on Hugging Face. This dataset contains all multiple-choice questions from the 2022, 2023, and 2024 editions of the Exame Nacional do Ensino Médio (ENEM), Brazil's national standardized exam used for university admissions.

To make the dataset suitable for evaluating textual-only models, the original image-based content from the exam booklets was replaced with accurate textual descriptions from the accessibility version (caderno laranja) of the ENEM. This allows large language models (LLMs) to process and respond to questions that originally included visual components.

To work with the dataset, I downloaded the JSONL file and loaded its contents line by line using Python’s json module. Each question entry includes the question text, multiple alternatives, and the correct answer label.

To facilitate subject-specific accuracy analysis, I assigned each question to one of the four official ENEM subject areas based on its position in the exam. This mapping follows the fixed structure used in every ENEM booklet:

* Questions 1–45: Linguagens, Códigos e suas Tecnologias (Languages)
* Questions 46–90: Ciências Humanas e suas Tecnologias (Human Sciences)
* Questions 91–135: Ciências da Natureza e suas Tecnologias (Natural Sciences)
* Questions 136–180: Matemática e suas Tecnologias (Mathematics)

This subject labeling was implemented by iterating through the dataset and adding a "subject" field to each entry accordingly. The final dataset used in the experiments thus consisted of 180 labeled questions from the 2024 ENEM exam, structured for consistent use across all prompting techniques.

## Run Tests

To evaluate the performance of the three prompting techniques — Chain-of-Thought (CoT), Chain-of-Verification (CoVe), and Self-Refine — an automated testing pipeline was developed so that it systematically applies each technique to the full ENEM 2024 dataset. The test loop runs each method over the dataset multiple times, capturing both the model outputs and relevant metadata for analysis.

In total, there are **38 valid samples per method** (i.e., 38 responses per question across runs).

Each test run executes a corresponding method-specific function — run\_cot, run\_cove, or run\_self\_refine — and stores the results in structured CSV files using save\_results\_csv() for subsequent evaluation.

**For Chain-of-Thought (CoT):**

* A 3-shot CoT prompt generates a reasoning-based answer for each question.
* The final answer letter is extracted from the response.
* Metadata recorded includes:
  + model response text,
  + predicted answer,
  + ground truth correctness,
  + question ID and subject label.

**For Chain-of-Verification (CoVe):**

* The model first generates a baseline answer using the CoT prompt.
* It then plans verification questions to check specific claims in the reasoning.
* These questions are independently answered, and a final revised answer is generated based on the verifications.
* Metadata includes:
  + the initial CoT answer and selected option,
  + the list of verification questions and their answers,
  + the final answer and correctness,
  + question ID and subject label.

**For Self-Refine:**

* The model generates an initial answer using the CoT prompt.
* It then iteratively critiques and refines its response in up to 10 feedback-revision loops.
* Iterations stop early if the selected answer remains unchanged.
* Metadata includes:
  + the initial and final answers,
  + the full sequence of intermediate answers,
  + the full trace of model responses and feedback,
  + correctness,
  + question ID and subject label.

All results were saved across runs, forming a dataset of structured responses enriched with metadata such as timestamps, trace steps, and reasoning sequences. This setup enabled robust analysis across different dimensions — including accuracy, consistency, and performance by subject — and supports statistical validation of the results.

## Clean Data

After running and storing the raw results from all test iterations, I applied a structured data cleaning pipeline to prepare the data for analysis. Each result file included the question ID, subject labels translated to English, ground truth and final answer, as well as the baseline answer for CoVe and Self-Refine. The cleaning process ensured consistency across methods and enabled accurate performance comparisons.

**Subject Normalization**  
Since the subject labels in the raw dataset were in Portuguese, they were mapped to their English equivalents to simplify analysis and visualization.

**Cleaning Process per Method**  
The process\_files() function iterated through all result CSV files for each method (CoT, CoVe, and Self-Refine). It detected new raw result files using a timestamped filename pattern and generated cleaned versions using method-specific cleaning functions. These cleaned datasets were saved to a separate directory with the suffix \_clean.

Each method had its own cleaning logic:

* **Self-Refine:** Extracted the initial answer either from the dedicated initial\_answer column or by parsing the baseline\_answer using a regular expression. The cleaned file retained id, translated subject, ground\_truth, predicted, correct, and initial\_answer.
* **CoVe:** Used the initial\_answer field directly from the raw output and retained the same core columns as Self-Refine.
* **CoT:** Focused solely on the final prediction and correctness, omitting the initial answer since no refinement is involved.

All cleaned DataFrames were saved as new CSV files, preserving only the relevant fields needed for the final analysis.

## Analyse Performance

To evaluate and compare the effectiveness of the prompting techniques, a quantitative analysis was performed based on three key metrics: overall accuracy, subject-wise accuracy, and consistency across runs. All analyses were conducted on the cleaned datasets generated for each of the 38 runs per method.

### General Accuracy

A general comparison of accuracy distributions for each method was conducted to address the **Research Question 1**:

**RQ1**: *How does the application of CoT, CoVe and Self-Refine affect LLM response accuracy on Brazilian University Entrance Exams?*

#### **Computation**

To measure overall accuracy, the cleaned results from all runs of each method were combined into a unified DataFrame. For each question, the correct field (a Boolean indicating whether the model selected the right answer) was used as the basis for the analysis. I first grouped the data by method and run to compute per-run accuracy, then aggregated the results to calculate the mean and standard deviation for each method across its 38 runs.

#### **First Results**

The following table shows the average accuracy across all ENEM questions for each technique, along with the standard deviation:

|  |  |  |  |
| --- | --- | --- | --- |
| **Method** | **Mean Accuracy** | **Standard Deviation** | **Runs** |
| CoT | 0.702485 | 0.190838 | 38 |
| CoVe | 0.671491 | 0.206063 | 38 |
| Self-Refine | 0.686257 | 0.193998 | 38 |

Table 1: General Mean Accuracy and Standard Deviation of CoT, CoVe and Self-Refine across test runs

#### **Statistical Analysis**

To assess whether the differences in overall accuracy between the three prompting methods — Chain-of-Thought (CoT), Chain-of-Verification (CoVe), and Self-Refine — were statistically significant, a series of statistical tests were conducted.

**Step 1: Data Aggregation**  
Each method was evaluated across 38 test runs. For each run, the average accuracy was computed per subject, resulting in a distribution of per-run correctness scores. This provided a balanced, run-level basis for statistical comparison.

**Step 2: Assumption Testing**  
Before applying a statistical test to compare methods, I tested whether the conditions for parametric analysis (such as ANOVA) were met:

* **Normality (Shapiro–Wilk test):**
  + CoT: W = 0.8965, p < 0.0001
  + CoVe: W = 0.9106, p < 0.0001
  + Self-Refine: W = 0.8944, p < 0.0001

All three distributions showed statistically significant deviation from normality (p < 0.05), indicating that the normality assumption was violated.

* **Homogeneity of Variances (Levene’s test):**
  + Statistic = 1.0745, p = 0.3423

The results of Levene’s test indicate that the variance across groups is not significantly different, satisfying the assumption of equal variances.

**Step 3: Non-Parametric Comparison (Kruskal–Wallis test)**  
Given the non-normality of the data, the **Kruskal–Wallis H test** was applied. It is a non-parametric alternative to ANOVA and compares the distributions of accuracy across the three methods.

The hypotheses were:

* **H₀ (Null Hypothesis):** The distributions of accuracy scores across the three prompting methods (CoT, CoVe, and Self-Refine) are the same.
* **H₁ (Alternative Hypothesis):** At least one method’s distribution differs significantly from the others.

This test does not assume normality and is suitable for comparing more than two groups.

* **H statistic**: 1.4336
* **p-value**: 0.4881

**Interpretation**  
Since the Kruskal–Wallis test resulted in a p-value of 0.4881, which is well above the conventional threshold of 0.05, we fail to reject the null hypothesis (H₀). This means that there is no statistically significant evidence to suggest that the three prompting methods differ in their accuracy performance across runs.

Although Chain-of-Thought (CoT) achieved the highest mean accuracy (70.2%), followed by Self-Refine (68.6%) and Chain-of-Verification (CoVe) (67.1%), these observed differences could plausibly be attributed to random variation. The lack of statistical significance indicates that none of the methods consistently outperformed the others in a way that generalizes across all evaluation runs.

This finding informs a nuanced answer to **Research Question 1 (RQ1)**: while prompting techniques like CoT and Self-Refine exhibit numerically higher performance, the statistical analysis does not support a definitive conclusion that they are more accurate than CoVe.

### Subject-Specific Accuracy

A subject-wise comparison of accuracy distributions for each method was conducted to address the **Research Question 2:**

**RQ2:** How does the effectiveness of Chain-of-Thought, Chain-of-Verification, and Self-Refine prompting techniques vary across different subject areas?

**Metric Aggregation**  
Accuracy scores were grouped by method, run, and subject, and the mean accuracy was calculated for each unique combination. This approach allowed for a consistent and comparable evaluation across the 38 runs of each method while preserving variability across academic domains.

A graph of different colored bars

AI-generated content may be incorrect.

Figure 1: Accuracy Mean and Standard Deviation per Method and Subject

**Hypotheses and Statistical Method**  
To determine whether the prompting methods differed significantly in accuracy for specific subjects, the **Kruskal–Wallis H test** was employed. This non-parametric test was selected due to violations of the normality assumption in the accuracy distributions, as established in the previous section.

For each subject, the following hypotheses were tested:

* **H₀ (Null Hypothesis):** The distributions of accuracy scores across the three prompting methods (CoT, CoVe, Self-Refine) are statistically indistinguishable.
* **H₁ (Alternative Hypothesis):** At least one method differs significantly in accuracy from the others.

**Results**  
The results of the Kruskal–Wallis tests, along with the average accuracy scores for each method and subject, are presented below:

Table 2: Average Accuracy by Subject and Kuskal-Wallis Results

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Subject** | **CoT Mean** | **CoVe Mean** | **Self-Refine Mean** | **Kruskal-Wallis H** | **p -value** | **p < 0.05** |
| Human Sc. | 0.954971 | 0.933918 | 0.942105 | 9.830247 | |  | | --- | | 7.334813e-03 | | True |
| Languages | 0.803509 | 0.786550 | 0.788304 | 4.642850 | |  |  | | --- | --- | | 9.813362e-02 |  | | False |
| Mathematics | 0.492398 | 0.413450 | 0.473684 | 35.809116 | |  |  | | --- | --- | | 1.675519e-08 |  | | True |
| Natural Sc. | 0.559064 | 0.552047 | 0.540936 | 4.270775 | 1.181988e-01 | False |

Statistically significant differences were observed for **Mathematics** and **Human Sciences**, indicating that at least one method outperformed the others within these subjects. In both cases, Chain-of-Thought achieved the highest average accuracy. No significant differences were found for **Languages** and **Natural Sciences**, suggesting comparable performance across methods for these disciplines.

**Interpretation**  
The findings indicate that prompting technique effectiveness is subject-dependent. The performance gap in **Mathematics**, a domain requiring logical precision and step-by-step reasoning, reinforces the advantages of approaches like Chain-of-Thought, which promote structured reasoning. Conversely, in subjects such as **Languages** and **Natural Sciences**, the influence of prompting format appears less pronounced, potentially due to the nature of the questions favoring comprehension over inference.

**Conclusion for RQ2**  
The analysis confirms that the choice of prompting strategy can impact LLM performance differently depending on the academic domain. These results support a positive answer to **RQ2**, emphasizing the importance of adapting prompting techniques to the subject matter to maximize accuracy and reliability.

### Consistency

Large Language Models are inherently stochastic, meaning the same prompt can lead to different outputs across runs. This section addresses **Research Question 3:**

**RQ3**: How consistent are the predictions of Chain-of-Thought, Chain-of-Verification, and Self-Refine methods across repeated runs?

A graph of different colored bars

AI-generated content may be incorrect.

Figure 2: Answer Consistency per Method and Subject

**Definition of Consistency**  
For this analysis, a prediction was marked as **consistent** if the model produced the same final answer for the same question across all 38 independent runs. Formally, consistency was computed by grouping predictions by question ID and counting how many unique answer choices were given. If only one answer appeared, the prediction was flagged as consistent.

**Overall Consistency**  
The table below shows the proportion of questions for which each method produced consistent answers across all runs:

|  |  |
| --- | --- |
| **Method** | **Consistency Rate** |
| CoT | 47.2% |
| CoVe | 29.4% |
| Self-Refine | 30.0% |

These results indicate that **Chain-of-Thought** produces the most stable predictions, followed by **Self-Refine** and **CoVe**.

McNemar’s test was selected to compare consistency between prompting methods because it is specifically designed for **paired binary data**. In this study, each question was evaluated by all methods, and each method's prediction was classified as either **consistent** or **inconsistent** across runs. This creates matched observations suitable for McNemar’s test.

The test assesses whether the **frequency of discordant outcomes** differs significantly between two methods, for example, whether one is more likely to produce consistent answers than the other.

**Hypotheses for McNemar’s Test**

* **H₀ (Null Hypothesis):** The consistency proportions of the two methods being compared are equal (no difference in consistency).
* **H₁ (Alternative Hypothesis):** The consistency proportions differ between the two methods.

**McNemar’s Tests for Overall Consistency**

|  |  |  |
| --- | --- | --- |
| **Comparison** | **p -value** | **Significant** |
| CoT vs CoVe | < 0.0001 | True |
| CoT vs Self-Refine | < 0.0001 | True |
| CoVe vs Self-Refine | 1.0000 | False |

The results show that **CoT is significantly more consistent** than both CoVe and Self-Refine (p < 0.0001), while there is no statistically significant difference between CoVe and Self-Refine (p = 1.0000).

**McNemar’s Tests for Consistency by Subject**  
To examine whether consistency varies by academic domain, consistency rates were calculated per subject area. The summary statistics are shown below:

|  |  |  |  |
| --- | --- | --- | --- |
| **Subject** | **Comparison** | **p -value** | **Significant** |
| Human Sciences | CoT vs CoVe | 0.0001 | True |
|  | CoT vs Self-Refine | 0.0020 | True |
|  | CoVe vs Self-Refine | 0.3438 | False |
| Languages | CoT vs CoVe | 0.0213 | True |
|  | CoT vs Self-Refine | 0.0004 | True |
|  | CoVe vs Self-Refine | 0.1460 | False |
| Mathematics | All comparisons | >0.06 | False |
| Natural Sciences | All comparisons | >0.3 | False |

**Interpretation**  
The consistency analysis reveals a clear pattern: **Chain-of-Thought yields more stable predictions** than both Chain-of-Verification and Self-Refine, both overall and within specific subject areas like Human Sciences and Languages. The differences are statistically significant and consistent with the idea that structured, step-by-step reasoning promotes not only accuracy but also **stability** in responses.

Conversely, in content-heavy or computationally complex domains such as Natural Sciences and Mathematics, the consistency advantage of CoT diminishes, potentially due to increased variability in how LLMs handle these question types.

**Conclusion for RQ3**  
The results affirm that prompting technique influences not only accuracy but also consistency. Chain-of-Thought demonstrates a clear reliability advantage, while Self-Refine and CoVe exhibit greater variability.

# Discussion

## Potentials

## Limitations

## Possible Improvements

# Bibliography

|  |  |
| --- | --- |
| [1] | J. Wei, X. Wang, D. Schuurmans, M. Bosma, F. Xia, E. Chi, Q. V. Le und D. Zhou, „Chain-of-thought prompting elicits reasoning in large language models,“ *Advances in neural information processing systems 35,* 2022. |
| [2] | S. Dhuliawala, M. Komeili, J. Xu, R. Raileanu, X. Li, A. Celikyilmaz und J. Weston, „Chain-of-verification reduces hallucination in large language models,“ arXiv preprint arXiv:2309.11495, 2023. |
| [3] | A. Madaan, N. Tandon, P. Gupta, S. Hallinan, L. Gao, S. Wiegreffe, U. Alon, N. Dziri, S. Prabhumoye, Y. Yang, S. Gupta, B. P. Majumder, K. Hermann, S. Welleck und e. al, „SELF-REFINE: Iterative Refinement with Self-Feedback,“ *Advances in Neural Information Processing Systems 36,* 2023. |
| [4] | D. Nunes, R. Primi, R. Pires, R. Lotufo und R. Nogueira, „Evaluating GPT-3.5 and GPT-4 Models on Brazilian University Admission Exams,“ arXiv2303.17003, 2023. |

|  |  |
| --- | --- |
| [1] | J. Wei, X. Wang, D. Schuurmans, M. Bosma, F. Xia, E. Chi, Q. V. Le und D. Zhou, „Chain-of-thought prompting elicits reasoning in large language models,“ *Advances in neural information processing systems 35,* 2022. |
| [2] | S. Dhuliawala, M. Komeili, J. Xu, R. Raileanu, X. Li, A. Celikyilmaz und J. Weston, „Chain-of-verification reduces hallucination in large language models,“ arXiv preprint arXiv:2309.11495, 2023. |
| [3] | A. Madaan, N. Tandon, P. Gupta, S. Hallinan, L. Gao, S. Wiegreffe, U. Alon, N. Dziri, S. Prabhumoye, Y. Yang, S. Gupta, B. P. Majumder, K. Hermann, S. Welleck und e. al, „SELF-REFINE: Iterative Refinement with Self-Feedback,“ *Advances in Neural Information Processing Systems 36,* 2023. |
| [4] | D. Nunes, R. Primi, R. Pires, R. Lotufo und R. Nogueira, „Evaluating GPT-3.5 and GPT-4 Models on Brazilian University Admission Exams,“ arXiv2303.17003, 2023. |

List of Figures

[Figure 1: Example of name and year printed on spine. **Error! Bookmark not defined.**](#_Toc330300567)

List of Tables

[Table 1: Schedule for “Applied Mathematics”. **Error! Bookmark not defined.**](#_Toc330300577)

List of Abbreviations

|  |  |
| --- | --- |
| WWW | World Wide Web |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

Documentation table of AI-based tools

|  |  |  |
| --- | --- | --- |
| **AI-based tools** | **Intended use** | **Prompt, source, page, paragraph...** |
| **DeepL Translate** | Translation of an article in English | Source (XXX), Chapter X on page X-X |
| **ChatGPT (4.0)** | Grammar and spelling | "Please list issues with spelling and grammar in the following text: ..." Entire document |
|  |  |  |

A: Heading of Appendix A

B: Heading of Appendix B