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 Introduction 5](#_Toc196474367)

[1.1 Motivation 5](#_Toc196474368)

[1.2 Tasks 5](#_Toc196474369)

[2 Methodology 6](#_Toc196474370)

[2.1 Ideal Solution 6](#_Toc196474371)

[2.2 Requirements 7](#_Toc196474372)

[2.3 Development Process 9](#_Toc196474373)

[2.3.1 Prepare Dataset 9](#_Toc196474374)

[2.3.2 Define Prompting Templates 9](#_Toc196474375)

[2.3.3 Test Prompting Techniques 10](#_Toc196474376)

[2.3.4 Evaluation and Data Collection 10](#_Toc196474377)

[2.4 Tests 10](#_Toc196474378)

[2.4.1 General Accuracy 10](#_Toc196474379)

[2.4.2 Accuracy by Subject 11](#_Toc196474380)

[2.4.3 Consistency 11](#_Toc196474381)

[2.5 Tools 11](#_Toc196474382)

[3 Solution 12](#_Toc196474383)

[3.1 Answering the Research Questions 12](#_Toc196474384)

[3.2 Prompting Techniques 13](#_Toc196474385)

[3.2.1 Chain-of-Thought (CoT) 13](#_Toc196474386)

[3.2.2 Chain-of-Verification (CoVe) 14](#_Toc196474387)

[3.2.3 Self-Refine 15](#_Toc196474388)

[3.3 Load QA dataset 16](#_Toc196474389)

[3.4 Run Tests 16](#_Toc196474390)

[3.5 Clean Data 17](#_Toc196474391)

[3.6 Analyse Performance 18](#_Toc196474392)

[3.6.1 General Accuracy 18](#_Toc196474393)

[3.6.2 Subject-Specific Accuracy 20](#_Toc196474394)

[3.6.3 Consistency 22](#_Toc196474395)

[3.6.4 Interesting Findings 25](#_Toc196474396)

[4 Discussion 26](#_Toc196474397)

[4.1 Potentials 26](#_Toc196474398)

[4.2 Limitations 26](#_Toc196474399)

[4.3 Possible Improvements 26](#_Toc196474400)

[5 Bibliography 26](#_Toc196474401)

[6 Figures 27](#_Toc196474402)

[7 Tables 28](#_Toc196474403)

[List of Abbreviations 29](#_Toc196474404)

[Documentation table of AI-based tools 30](#_Toc196474405)

[A: Heading of Appendix A 31](#_Toc196474406)

[B: Heading of Appendix B 32](#_Toc196474407)

# Introduction

## Motivation

As a Computer Science student in the age of artificial intelligence, I have often turned to Large Language Models (LLMs) like ChatGPT to support my study routine. These tools act as accessible tutors, capable of answering questions, explaining concepts in simpler terms, and adapting to my individual learning pace. Especially attractive is the LLM’s ability to respond without judgment, enabling students like me to ask "dumb" questions, repeat topics as often as needed, and receive immediate, tailored feedback.

However, my positive experiences with LLMs were often met with a critical limitation: the **inconsistency and unreliability of factual information**. While LLMs often produce helpful answers, I noticed that some explanations were factually incorrect or misleading. In some cases, the error was obvious; in others, I would only recognize it after consulting external sources, raising the concern that I might be unintentionally **learning and reinforcing misconceptions.**

This observation reflects a broader problem: **LLMs can hallucinate**, producing outputs that sound plausible, but are factually incorrect [1]. This becomes a significant risk in educational settings, where students might not yet have the background knowledge to detect such inaccuracies. Students are left vulnerable to absorbing incorrect information with full confidence.

This concern is the central motivation of my thesis: **how can we reduce hallucinations in LLMs in a way that is simple and accessible to students?** While many solutions exist, such as fine-tuning models or incorporating retrieval systems, they are not easily accessible for a typical LLM chatbot user. Therefore, I focused on **prompt-based techniques** that require no special infrastructure, training, or external tools, and can be used with standard LLM interfaces like OpenAI's ChatGPT.

The use case I address is straightforward: students using LLMs to help them solve high-school level multiple-choice questions. If prompting techniques can reduce hallucinations and improve the reliability of answers in this setting, they would offer a **practical way to improve LLM-assisted learning** without requiring any technical expertise or system modifications from the user.

## Tasks

The primary goal of this thesis is to **evaluate the effectiveness of prompt-based techniques in reducing hallucinations in Large Language Models (LLMs)**, with a focus on scenarios where students use these models as learning aids. Instead of relying on model retraining or external tools, I investigate whether **LLM response** reliability can be improved using prompt engineering alone.

To this end, I aim to answer the following research 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 these prompting techniques vary across different subject areas?*

**RQ3**: *How consistent are the predictions of each method across repeated runs?*

These questions are explored through a **quantitative experiment** using a standardized high-school level exam (ENEM 2024) with multiple-choice questions across different subjects. This setup allows for automatic LLM response evaluation and comparison across methods.

The main outcome of this work is a **reproducible evaluation framework** implemented in Python using Jupyter Notebooks. This framework loads the questions dataset and applies each prompting strategy using the OpenAI API. It then collects results across multiple test runs, and calculates key performance metrics such as accuracy, subject-wise performance, and consistency. All code, prompt templates, and evaluation logic were designed with modularity and transparency in mind, making it straightforward to extend the approach to other datasets or prompting techniques in future studies.

By providing empirical insights and a practical testing pipeline, this thesis aims to contribute to ongoing efforts in **making LLM-assisted learning more trustworthy.**

# Methodology

## Ideal Solution

An evaluation of prompting techniques for Large Language Models (LLMs) requires a controlled and repeatable experimental setup that allows for fair 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, verified questions across multiple disciplines, ideally with rich information 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 limited by several factors, including API access constrains 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. Response consistency was estimated with results from multiple test runs, and the accuracy difference between prompting techniques was validated using statistical tests.

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

To explore effective hallucination mitigation strategies, I began by conducting a literature review focused on **state-of-the-art techniques** proposed in recent research. This overview of hallucination mitigation methods (see Figure 1) categorizes them into two main branches: Prompt Engineering and Developing Models. While model-based techniques like fine-tuning require infrastructure and model training expertise beyond the reach of typical users, prompt engineering methods stand out for their accessibility and low entry barrier. These methods can be applied directly using commercial APIs and chatbots, making them ideal for scenarios where students interact with LLMs as black-box systems.

A diagram of a company

AI-generated content may be incorrect.

Figure 1: Imama, “Beyond Traditional Fine-tuning: Exploring Advanced Techniques to Mitigate LLM Hallucinations,” Hugging Face Blog, Apr. 2024. [Online]. Available: <https://huggingface.co/blog/Imama/pr>. [Accessed: Apr. 10, 2025].

Within prompt engineering, three sub-categories are outlined: ***Retrieval-Augmented Generation (RAG)*, *Prompt Tuning***, and ***Self-Refinement through Feedback and Reasoning***. Based on feasibility, I excluded the first two: RAG relies on external knowledge bases or search modules [2], which a student may not have access to, and Prompt Tuning requires gradient-based optimization and training tokens [3], which is impractical for everyday users.

This narrowed the focus to **self-refinement methods**: a subset of prompt engineering that enhances LLM performance through structured reasoning and self-improvement. Among the techniques surveyed, I identified three suitable candidates that are both effective and feasible:

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

Encourages models to produce intermediate reasoning steps, improving performance on complex multi-step problems [4].

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

Adds a verification phase that critically assesses and revises the model’s initial answer, shown to reduce hallucinations [5].

**SELF-REFINE**:

Implements an iterative loop of feedback and refinement, helping models self-correct reasoning and improve factual accuracy [6].

These techniques were selected because they:

1. Require no model fine-tuning or external tools,
2. Operate entirely through prompting, and
3. Can be reproduced with standard access to the OpenAI API.

The design of the experiment was also shaped by the **evaluation needs of these techniques**. To assess factual correctness, the **ENEM 2024 university entrance exam** [7] **was selected. It contains 180 multiple-choice questions across four subject domains, along with the ground-truth answers**. This allowed for straightforward, automated scoring without the need to judge the reasoning quality, ideal for isolating the effects of prompt design alone.

Finally, the **evaluation metrics and prompt template structures** were based on the goals of the selected methods and the prior literature. Accuracy, subject-specific performance, and consistency were chosen to capture performance, domain sensitivity, and stability, respectively, aligning with how prior work evaluated CoT [4], CoVe [5], and Self-Refine [6]. Structured prompt templates were created for each method for consistency across test runs while enabling method-specific logic such as verification or refinement loops.

Together, these requirements ensured that the study would not only provide insight into the effectiveness of self-refinement prompting methods, but also provide actionable results applicable in real-world student use cases.

## 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.

### Prepare Dataset

The experiments were conducted using the publicly available ENEM 2024 exam dataset [7], which contains all multiple-choice questions from the 2024 Brazilian national university entrance exam (Exame Nacional do Ensino Médio). A total of **180 questions** were used, covering four official subject areas:

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

Each entry 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 based on question order, reflecting the standard structure of ENEM exams.

### Define Prompting Templates

To ensure consistency and control over input formatting, **custom templates** were created for each of the three prompting 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 verification questions before revising them for the final answer.

**Self-Refine:**

Combined CoT 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.

### Test Prompting Techniques

The execution pipeline was implemented in a **Jupyter Notebook** using the GPT-3.5 model via OpenAI API. Each question was run **40 times per method** to capture variability and analyse consistency. Each response was recorded along with metadata such as timestamps, reasoning traces, and intermediate answers.

### 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,
* The subject label,
* Additional metadata depending on the method:
  + **CoVe:** Verification questions and answers
  + **Self-Refine:** Iterative responses, feedback trace and intermediate answers

This structured dataset enabled an 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

### 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 40 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 ENEM subjects. For every method and run, subject-specific accuracy was calculated as the mean correctness over questions in that subject area. These 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 40 runs. The overall consistency rate was computed as the proportion of questions classified as consistent.

To analyse 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 templates.

**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 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 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 prompting method was implemented as a reusable prompt template and applied to the same multiple-choice 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.

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

## Prompting Techniques

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

**Definition**

Chain-of-Thought (CoT) is a prompting technique designed to improve the reasoning capabilities of 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. As reported in *Chain-of-Thought Prompting Elicits Reasoning in Large Language Models* [4] , 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-step process:

1. **Instructions**:

The model is introduced to the task through a 3-shot prompt containing example ENEM questions and detailed explanations. It is then presented with the question it should solve by following the examples.

1. **Generate reasoning**:

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**

CoT was implemented using a Python prompt template tailored for ENEM-style multiple-choice questions. Each prompt included three examples taken from *Evaluating GPT-3.5 and GPT-4 Models on Brazilian University Admission Exams* [7], where the authors evaluated GPT-3.5 and GPT-4 models on ENEM exams. These examples were selected from the **ENEM 2022 exam** and span across three subject areas: **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 the options, and a final line that explicitly states the correct alternative in a structure format. The explanations were derived from expert teacher discussions and public exam commentary resources, as also documented in the paper [7].

In the prompt template, the 3-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 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.

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

**Definition**

Chain-of-Verification (CoVe) is a prompting method developed to reduce hallucinations in LLMs by having the model **verify its own outputs**. Introduced in *Chain-of-Verification Reduces Hallucination in Large Language Models* [5], CoVe operates on the assumption that LLMs can critically analyse and improve their responses when prompted appropriately. Rather than relying on a single pass, CoVe decomposes the generation process into separate verification steps.

**Key Steps in the Process**

The CoVe process involves **four sequential stages**:

1. **Generate a Baseline Response:**  
   An initial draft is generated using Chain-of-Thought template.
2. **Plan Verification Questions:**  
   The model reflects on the baseline response and produces verification questions to check for factual correctness or reasoning steps.
3. **Execute Verifications:**  
   Each verification question is answered in isolation, preventing 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, refined answer.

**Implementation for ENEM Questions**

In this study, CoVe was implemented as a three-step pipeline. All steps are executed inside an iterative loop that repeats the CoVe pipeline over the ENEM questions.

The process begins by generating a **baseline response** using the 3-shot Chain-of-Thought template described in the CoT section. This response serves as the initial answer.

The second step uses the plan\_verification\_questions() function to generate verification questions. The prompt instructs the LLM to reflect on its previous answer and identify key factual or logical claims worth verifying.

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​ [5].

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** [5].

### ****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 has shown to improve LLM performance across diverse tasks, including reasoning, coding, and factual QA [6].

**Key Steps in the Process**

The implementation consisted of three steps:

1. **Initial Answer**:

The model answers the ENEM question using the Chain-of-Thought template, producing an initial answer.

1. **Feedback**:

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 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 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.

## Load QA dataset

The question-answer dataset used in this study was sourced from the open-access Maritaca AI ENEM dataset [7], hosted on Hugging Face. It contains all multiple-choice questions from the 2024 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 textual descriptions from the accessibility version (caderno laranja) of the ENEM. This allows 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 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.

## Run Tests

To evaluate the performance of Chain-of-Thought (CoT), Chain-of-Verification (CoVe), and Self-Refine, a testing pipeline was developed so that it systematically applies each technique to the full ENEM 2024 dataset. For each method there is a total of 40 responses per question.

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.

## Clean Data

After running and storing the raw results from all test iterations, I applied a structured data cleaning pipeline to extract the relevant fields for analysis. Each result file included the question ID, subject labels translated to English, ground truth and final answer, as well as the initial answer for CoVe and Self-Refine.

**Subject Translation**  
Since the subject labels in the original 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, a correct flag, and initial\_answer.

**CoVe:**

Used the initial\_answer field directly from the raw output and retained the same columns as Self-Refine.

**CoT:**

Focused solely on the final prediction and correctness, since no refinement is involved.

## Analyse Performance

To 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 40 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 single DataFrame. For each question, the correct flag 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 40 runs.

A graph of different colored bars

AI-generated content may be incorrect.

Figure 2: Overall Accuracy by Method

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 | 0.703750 | 0.189968 | 40 |
| CoVe | 0.672917 | 0.205645 | 40 |
| Self-Refine | 0.685694 | 0.193422 | 40 |

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

#### **Statistical Analysis**

The next step was to assess whether the differences in overall accuracy between the three methods were statistically significant.

**Testing for Normality and Homogeneity**  
I first tested whether the conditions for parametric analysis, such as ANOVA, were met:

* **Normality (Shapiro–Wilk test):**
  + CoT: W = 0.8983, p < 0.0001
  + CoVe: W = 0.9108, p < 0.0001
  + Self-Refine: W = 0.8929, 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 = 1.2022, p = 0.3014

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

**Non-Parametric Comparison (Kruskal–Wallis test)**  
Given the non-normality of the data, the **Kruskal–Wallis H test** [8] was applied to compare the distributions of accuracy across the three methods.

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

**Interpretation**

* **H statistic**: 3.7021
* **p-value**: 0.1570

The Kruskal–Wallis test resulted in a p-value of 0.1570, which is above the conventional threshold of 0.05, so we **fail to reject the null hypothesis (H₀)**. The lack of statistical significance indicates that none of the methods consistently outperformed the others in a way that generalizes across all evaluation runs. Although Chain-of-Thought achieved the highest mean accuracy (70.3%), followed by Self-Refine and Chain-of-Verification (67.3%), these observed differences could be attributed to random variation.

**Conclusion for RQ1**

This finding brings an answer to **Research Question 1 (RQ1)**: while CoT exhibits higher performance, the statistical analysis does not support a definitive conclusion that it is more accurate than CoVe or Self-Refine.

### Subject-Specific Accuracy

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

**RQ2:** How does the effectiveness of Chain-of-Thought, Chain-of-Verification, and Self-Refine 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.

A graph of different colored bars

AI-generated content may be incorrect.

Figure 3: Accuracy Mean and Standard Deviation per Method and Subject

**Hypotheses and Statistical Method**

**Conside**ring the violations of the normality assumption in the accuracy distributions, as established in the previous section, the **Kruskal–Wallis H test** was employed to determine whether the prompting methods differed significantly in accuracy for specific subjects.

* **H₀ (Null Hypothesis):** The distributions of accuracy scores across the three prompting methods are statistically indistinguishable for any subject.
* **H₁ (Alternative Hypothesis):** At least one method differs significantly in accuracy from the others for a given subject.

**Results**

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 subjects.

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. | 0.955000 | |  |  | | --- | --- | | 0.934444 |  | | 0.942222 | 10.425922 | |  |  |  | | --- | --- | --- | | |  |  | | --- | --- | | 5.445527e-03 |  | | | True |
| Languages | 0.803889 | |  |  | | --- | --- | | 0.787778 |  | | |  |  | | --- | --- | | 0.785556 |  | | 5.508679 | |  |  |  |  | | --- | --- | --- | --- | | |  |  | | --- | --- | | 6.365103e-02 |  | |  | | False |
| Mathematics | 0.495000 | 0.415000 | 0.475000 | |  |  | | --- | --- | | 37.652330 |  | | |  |  | | --- | --- | | 6.666540e-09 |  | | True |
| Natural Sc. | |  |  | | --- | --- | | 0.561111 |  | | 0.554444 | |  |  | | --- | --- | | 0.540000 |  | | 5.743476 | |  |  | | --- | --- | | 5.660046e-02 |  | | False |

**Interpretation**  
For **Mathematics** and **Human Sciences**, the p-values are both below the significance threshold of 0.05. Therefore, we **reject the null hypothesis (H₀)** for Mathematics and Human Sciences, as there are statistically significant differences in accuracy between the prompting methods. In both of these subjects, **Chain-of-Thought achieved the highest mean accuracy** (Mathematics: 0.495, Human Sciences: 0.955), indicating it outperformed both Chain-of-Verification and Self-Refine.

For **Languages** (p = 0.06365103) and **Natural Sciences** (p = 0.05660046), the p-values are above the 0.05 threshold. Therefore, we **fail to reject the null hypothesis (H₀)** for these subjects, indicating that there is no statistically significant difference in accuracy between the three prompting methods.

**Conclusion for RQ2**

**These findings** bring an answer to Research Question 2 (RQ2): there was a significant difference between the methods’ performance in Mathematics and Human Sciences, with Chain-of-Thought offering the most benefit in both cases. In subjects like Languages and Natural Sciences, all methods perform similarly.

### Consistency

Large Language Models are 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 with different colored bars

AI-generated content may be incorrect.

Figure 4: 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 40 test runs.

**Overall Consistency**

|  |  |
| --- | --- |
| **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**.

To compare consistency between prompting methods, the McNemar’s test [9] was selected. The test is designed for **paired binary data**, and in this study, each method's prediction was classified as either **consistent** or **inconsistent** across runs for a given question. 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**

* **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**

|  |  |  |  |
| --- | --- | --- | --- |
| **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**  
Based on the test results, we **reject the null hypothesis** (**H₀)** for comparisons involving Chain-of-Thought (CoT) versus both Chain-of-Verification (CoVe) and Self-Refine, indicating that **CoT is significantly more consistent** overall (p < 0.0001). However, we **fail to reject the null hypothesis** for CoVe versus Self-Refine (p = 1.0000), suggesting no significant difference in their consistency.

At the subject level, CoT also shows significantly higher consistency than both CoVe and Self-Refine in **Human Sciences** and **Languages**, but no significant differences were observed in **Mathematics** and **Natural Sciences**. These results support the alternative hypothesis (H₁) for **Human Sciences** and **Languages domains.**

**Conclusion for RQ3**

**These results** answer Research Question 3 (RQ3): **Chain-of-Thought** produces significantly more consistent predictions than both **Chain-of-Verification** and **Self-Refine**, both overall and within specific subject areas like **Human Sciences** and **Languages.**

### Interesting Findings

A closer examination of the subject-wise accuracy gains reveals insights about the effectiveness of feedback-based techniques in this context. While Self-Refine and CoVe are designed to improve upon initial answers through iterative reasoning or verification, results show that this is not always the case.

A test was performed to assess what would happen if both methods had always stuck with their initial answer: the output produced using the Chain-of-Thought template, before any feedback or revision. Surprisingly, in several cases, this **baseline answer proved to be statistically more accurate than the final refined one**.

A graph with green and blue bars

AI-generated content may be incorrect.

Figure 5: Accuracy gain by sticking with the initial answer

Table 3: Statistical Analysis of Initial vs Final Answer Accuracy for CoVe and Self-Refine

A screenshot of a test

AI-generated content may be incorrect.

**CoVe**

For CoVe, the final refinement underperformed the initial CoT answer in **three out of four subjects** (Human Sciences, Languages, and Mathematics) with statistically significant drops in accuracy (p < 0.001 for Mathematics, for example). The only exception was in Natural Sciences, where the difference was not statistically significant. This pattern suggests that **CoVe’s verification mechanism may sometimes introduce errors or reinforce uncertainties** rather than resolving them, possibly due to flawed verification questions or overly cautious revisions.

**Self-Refine**

Self-Refine showed a consistent trend of **accuracy decrease after refinement** in most subject areas. In **Languages, Mathematics,** and **Natural Sciences**, the final answer generated after the feedback–refinement loop was **significantly worse** than the initial response generated using the Chain-of-Thought template (p < 0.001 for both Languages and Mathematics; p = 0.01 for Natural Sciences). These results suggest that, rather than improving the original answer, the **iterative feedback mechanism of Self-Refine often introduced noise or instability that led to poorer outcomes**.

**Conclusion**

Overall, these results highlight a **critical insight: more reasoning steps do not always lead to more accurate answers**. In fact, for certain subjects and methods, **the best result was already achieved in the first answer,** raising questions about when and how refinement should be applied. It suggests that a hybrid strategy, where refinement is selectively applied based on initial confidence or answer quality, may be a more effective approach in educational tasks.

# Discussion

## Potentials

## Limitations

## Possible Improvements

# Bibliography

|  |  |
| --- | --- |
| [1] | Z. Ji, N. Lee, R. Frieske, T. Yu, D. Su, Y. Xu, E. Ishii, Y. Bang, D. Chen, W. Dai, H. S. Chan, A. Madotto and P. Fung, "Survey of Hallucination in Natural Language Generation," ACM computing surveys 55, 2023. |
| [2] | P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, S. Riedel, D. Kiela and L. Zettlemoyer, "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks," *Advances in Neural Information Processing Systems (NeurIPS),* 2020. |
| [3] | B. Lester, R. Al-Rfou and N. Constant, "The Power of Scale for Parameter-Efficient Prompt Tuning," arXiv preprint arXiv:2104.08691, 2021. |
| [4] | J. Wei, X. Wang, D. Schuurmans, M. Bosma, F. Xia, E. Chi, Q. V. Le and D. Zhou, "Chain-of-thought prompting elicits reasoning in large language models," *Advances in neural information processing systems 35,* 2022. |
| [5] | S. Dhuliawala, M. Komeili, J. Xu, R. Raileanu, X. Li, A. Celikyilmaz and J. Weston, "Chain-of-verification reduces hallucination in large language models," arXiv preprint arXiv:2309.11495, 2023. |
| [6] | 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 and e. al, "SELF-REFINE: Iterative Refinement with Self-Feedback," *Advances in Neural Information Processing Systems 36,* 2023. |
| [7] | D. Nunes, R. Primi, R. Pires, R. Lotufo and R. Nogueira, "Evaluating GPT-3.5 and GPT-4 Models on Brazilian University Admission Exams," arXiv2303.17003, 2023. |
| [8] | W. H. Kruskal and W. A. Wallis, "Use of ranks in one-criterion variance analysis," *Journal of the American statistical Association,* vol. 47, no. 260, pp. 583-621, 1952. |
| [9] | Q. McNemar, "Note on the sampling error of the difference between correlated proportions or percentages," *Psychometrika,* p. 153–157, 1947. |

# Figures

[Figure 1: Overview of Hallucination Mitigation Techniques in LLMs 8](#_Toc195175595)

[Figure 2: Overall Accuracy by Method 19](#_Toc195175596)

[Figure 3: Accuracy Mean and Standard Deviation per Method and Subject 21](#_Toc195175597)

[Figure 4: Answer Consistency per Method and Subject 23](#_Toc195175598)

# Tables

[Table 1: General Mean Accuracy and Standard Deviation of CoT, CoVe and Self-Refine across test runs 19](#_Toc195175846)

[Table 2: Average Accuracy by Subject and Kuskal-Wallis Results 21](#_Toc195175847)

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