Degree Project in Computer Science and Engineering

Second cycle, 30 credits

Mitigating Hallucination in Large Language Model Code Generation for Higher Education

An Evaluation of Retrieval Augmented Generation

ALI ASBAI

Mitigating Hallucination in Large Language Model Code Generation for Higher Education

An Evaluation of Retrieval Augmented Generation

ALI ASBAI

Master's Programme, Computer Science, 120 credits
Date: June 18, 2024

Supervisor: Olga Viberg
Examiner: Olov Engwall
School of Electrical Engineering and Computer Science
Swedish title: Mitigering av Hallucination i Stora Språkmodellers Kodgenerering inom Högre Utbildning
Swedish subtitle: En Utvärdering av Retrieval Augmented Generation

Abstract

Generative AI has gained a rapidly increasing focus in programming higher education, where the broad spectrum of subjects covered by modern Generative AI tools has made them a vital source of accurate and dependable information. However, a critical concern with Generative AI lies in its tendency to produce inaccurate or fictional information. This tendency, often termed "hallucination", is a significant challenge for Generative AI. Especially in the field of programming higher education since it can lead to students' inaccurate knowledge acquisition and ultimately affect their learning process negatively. The present study evaluated one of the popular hallucination mitigation techniques, namely Retrieval Augmented Generation on ChatGPT and Gemini. In order to maximise their proficiency in solving programming assignments across varying difficulty levels. An experiment was conducted where ChatGPT and Gemini were tested on 100 randomly selected programming problems provided by Kattis, an automatic software grading tool for computer science programs often used in higher education. ChatGPT and Gemini attempted to solve each assignment, both not using Retrieval Augmented Generation, and using it, the solutions generated were automatically evaluated by Kattis. The results demonstrated an average 10% increase in acceptance rates for both models when using Retrieval Augmented Generation. Furthermore, using Retrieval Augmented Generation led to a lower error rate as well as a higher rate of improvement on previous attempts. The results contribute to the ongoing research on the utility of Generative AI-powered tools in programming education.

Keywords

Generative Artificial Intelligence, Large Language Model, Retrieval Augmented Generation, Hallucination, ChatGPT, Gemini, Kattis, Programming Higher Education

Sammanfattning

Generativ AI har fått ett stort fokus inom högre utbildning för programmering, där de moderna Generativ AI-verktygen, på grund av det breda spektrumet av ämnen de täcker, har blivit en viktig källa för korrekt och pålitlig information. Dock finns det stor oro över Generativ AIs tendens att producera felaktig eller fiktiv information. Denna tendens, ofta kallad ”hallucination”, utgör en betydande utmaning för Generative AI inom programmeringsutbildning eftersom den kan leda till att studenterna erhåller felaktiga kunskaper och påverka deras inlärningsprocess negativt. Denna studien utvärderade hallucinationsbegränsningstekniken kallad Retrieval Augmented Generation på ChatGPT och Gemini för att maximera deras effektivitet i att lösa programmeringsuppgifter över olika svårighetsnivåer. Ett experiment utfördes där ChatGPT och Gemini testades på 100 slumpmässigt valda programmeringsproblem uppladade på Kattis, ett automatiskt mjukvarubetygssystem för datavetenskapsprogram som ofta används inom högre utbildning. ChatGPT och Gemini försökte lösa varje uppgift både med och utan Retrieval Augmented Generation, och de genererade lösningarna utvärderades av Kattis. Resultaten visar en ökad medel acceptansgrad på 10% för båda modellerna vid användning av Retrieval Augmented Generation. Dessutom ledde användningen av Retrieval Augmented Generation till en lägre felprocent samt en högre förbättringsgrad jämfört med tidigare försök. Resultaten bidrar till den pågående forskningen om användbarheten av Generative AI-drivna verktyg inom programmeringsutbildningar.

Nyckelord

Generativ Artificiell Intelligens, Stora Språkmodeller, Retrieval Augmented Generation, Hallucination, ChatGPT, Gemini, Kattis, Högre Utbildning inom Programmering

Acknowledgments

I would like to thank Olga Viberg for her incredible help both before and during my work on this thesis. Her support in conducting the experiment, along with her great feedback while writing the thesis, was key to my completing my work. I am also grateful to KTH for the outstanding education I have received during my five years here.

Stockholm, June 2024
Ali Asbai

Contents

List of Figures

List of Tables

List of acronyms and abbreviations

AI	Artificial Intelligence
AIGCD	AI-Generated Code Detection
Gen AI	Generative Artificial Intelligence
LLM	Large Language Model
RAG	Retrieval Augmented Generation

Chapter 1

Introduction

With the launch of OpenAIs ChatGPT in late 2022, Generative Artificial Intelligence (Gen AI) has gained increased focus in research and development in various sectors of society [1]. One such sector is programming higher education, where this technology has shown the ability to influence students' learning by improving students' computational thinking skills, programming self-efficiency, and motivation [2].

Publicly available Large Language Models (LLMs) such as ChatGPT and Gemini have amassed large popularity since their releases [3, 4]. They have provided students with a free and easy-to-use method to utilize Gen AI-powered tools to generate code [5]. This fact, combined with their widespread use among students [6], presents an opportunity to improve learning conditions in many ways. Such as Artificial Intelligence (AI)-generated assignments, AI-generated feedback, AI-generated explanations for complex subjects, as well as helping students overcome writers' block [7].

While previous research about AI-powered education tools prove their potential in supporting students [5, 7, 8, 9], research also shows the inconsistency and ineffectiveness of LLMs, such as for example, ChatGPT and Gemini, when solving coding tasks in the context of higher education, especially at advanced difficulty levels [10, 11, 12]. Therefore, the potential of the technology to support programming higher education is overshadowed by its inconsistency in solving programming problems meant for higher education since students and educators need to be confident in the accuracy of any information generated by these models.

1.1 Problem

A critical concern with LLMs lies in their tendency to produce inaccurate or fictional details concerning real-world subjects. This phenomenon, often termed ”hallucination”, has become a considerable challenge for Gen AI technology in the case of both text and code generation [13, 14, 15]. The cause of this issue stems from the pattern generation techniques employed during the training phase and the lack of real-time updates of the models' knowledge base [15]. The widespread use of Gen AI in higher education [6] makes LLMs' hallucinations an urgent obstacle since it may lead to the students' inaccurate knowledge acquisition and ultimately affect their learning process negatively.

Apart from changes in the creation of LLMs, different prompt engineering techniques to tackle the issue of hallucination have been introduced recently [16, 17, 18]. Retrieval Augmented Generation (RAG) [19] is one such technique; it effectively mitigates the issue of hallucination in LLMs by producing responses that are not just relevant and up-to-date but also verifiable [16]. Research into the effects of using RAG for text generation has shown great results in effectively assimilating external knowledge [20], boosting overall performances and accuracy of LLMs [17], as well as performing highly in several benchmarks [16]. However, to our knowledge, little research has been performed on using RAG for mitigating hallucination in LLMs when generating code.

1.2 Purpose

This study evaluated the RAG hallucination mitigation strategy on ChatGPT and Gemini to maximise their proficiency in solving programming tasks across various difficulty levels, from easy to difficult ones. More specifically, the study evaluated the RAG techniques' ability to mitigate hallucination in the setting of programming higher education. This was achieved by the utilization of *Kattis* [21], a widely adopted automated assessment tool in programming education, to evaluate the effectiveness of the RAG method. The research questions of this study are:

- **RQ1:** To what extent can ChatGPT and Gemini accurately solve programming assignments in the setting of higher education?

- **RQ2:** To what extent can prompt engineering through RAG mitigate hallucination in ChatGPT and Gemini code generation?

1.3 Goals

Examining the capabilities of popular LLMs involves assessing the range and effectiveness of the tasks they can perform. By understanding these capabilities, an evidence-based gauge of the potential that Gen AI tools have to enhance the learning conditions for programming students is acquired.

Therefore, to answer the research questions of this study, four sub-goals were formulated:

1. A significant number of programming assignments of various difficulties need to be chosen.
2. The capabilities of ChatGPT and Gemini must be examined by generating code for the programming assignments without the use of RAG or any other prompt engineering techniques.
3. RAGs' capabilities need to be examined by optimising the input prompts for ChatGPT and Gemini when generating code for the programming assignment.
4. The results of both runs need to be compared and analysed.

1.4 Structure of the thesis

Chapter 2 of this thesis provides a description of the background information relevant to the study. Chapter 3 introduces the dataset and methods that were used in this study. Lastly, the study's results are presented in Chapter 4 and further discussed in Chapter 5.

Chapter 2

Background

This chapter starts with explaining the role of Gen AI in higher education and programming higher education specifically. Secondly, this chapter provides a summary of the issue of hallucination in LLMs and the different mitigation strategies that researchers have proposed. Finally, this chapter introduces and examines previous research on the topics relevant to this thesis.

2.1 Generative AI in higher education

Gen AI has garnered significant attention and sparked discussions in the setting of higher education [22, 23, 24]. Differing viewpoints have surfaced regarding the technology's impact and influence, with some highlighting the potential advantages of integrating Gen AI tools in higher education to enrich the learning experience, while others expressed reservations about the possible repercussions [22, 23].

The empirical research on using Gen AI in higher education is growing, but the results are still limited. M. Montenegro-Rueda *et al.* [25] reviewed twelve papers about the topic and found that most of the research done by the time of the paper was theoretical in its aim to understand the potential and challenges of the tool. However, some studies used qualitative and quantitative methodologies to explore its effects. The main result of [25] was that Gen AI shows promise in transforming teaching and learning methods by enhancing performance, motivation, and time management and promoting more effective collaborative learning. However, the results also show that proper training of teachers and students is essential to ensure the correct usage of the tool in

an academic setting. Ethical implementation and supervision of the tool are therefore crucial to avoid misuse.

2.1.1 Educators attitude towards Generative AI

Research about educators' attitudes towards Gen AI has shown mixed results. It has been reported that most faculty members express caution towards the technology, with the potential risk of cheating and plagiarism being seen as the biggest roadblocks to overcome for the technology [22]. Another concern is the possibility of LLMs generating misleading or inaccurate information [26, 27]. Additionally, teachers express concern about the superficial nature of modern LLMs responses, citing that they may exhibit logical inconsistencies and incoherence that is dangerous for students lacking the necessary knowledge and skills to interact critically with it [28, 29]. On the other hand, there is research that claims that university lecturers generally hold a positive attitude towards integrating the use of Gen AI in teaching and learning [30, 31]. Faculty that have expressed a more positive attitude towards Gen AI are citing its potential in tasks such as automated feedback and grading [22] as well as helping lecturers in creating course materials efficiently [32, 33], which would give teachers more time to focus on the students. These teachers do not see Gen AI as a threat to the educational system as long as the data generated by the LLMs is verifiable [34].

2.1.2 Students attitude towards Generative AI

The students are another critical stakeholder who may benefit from or be challenged by Gen AI tools in their educational practices. Students generally recognise the potential of Gen AI to enhance their learning experiences and academic outcomes in various ways [23], such as access to immediate assistance personalised to their individual learning needs, brainstorming, feedback on writing, as well as research and analysis support through text summarisation [23]. However, students have also expressed concerns regarding the use of Gen AI in higher education. These concerns mainly revolved around the accuracy of AI-generated information, privacy issues in fear that AI technologies may collect personal data and ethical issues connected to the risk of plagiarism in AI-generated text [23]. Besides these concerns, students have also expressed reservations about becoming overly

dependent on Gen AI tools since this might hinder the development of holistic competencies, such as students' critical thinking and creativity, compared to traditional education. This, in turn, can negatively impact students' future career opportunities [23].

2.1.3 Programming Education

Computer programming has become an important skill across various industries in today's modern business landscape. Possessing programming skills empowers individuals to pioneer and develop innovative technologies, fueling innovation and economic expansion. Having computer programming skills can give individuals "the ability to create and build new technologies that can drive innovation and economic growth" [2]. Therefore, employers value individuals capable of programming when hiring. This extends beyond the technology sector to all rapidly digitised fields such as finance, health, transportation, and education [2]. Consequently, educational institutions are adapting to the evolving demands of today's business landscape, resulting in a widespread provision of programming education, ranging from early childhood to higher education [35]. Computer programming is the foundation of the digital world, and its significance only grows over time. Proficiency in programming empowers individuals to understand how technology functions as well as their potential applications and exploitation. Additionally, computer programming enhances problem-solving and critical thinking skills [36]. Computer programming offers a structured method for expressing concepts and addressing issues with applications across various domains. Proficiency in coding, regardless of one's profession, equips individuals with problem-solving skills that can enhance their effectiveness in various aspects of life [37].

Higher education in programming involves dealing with abstract concepts and logic, which can be challenging for individuals more accustomed to concrete tasks [38]. This challenge creates a steep learning curve that is critical to overcome in a world where programming skills and knowledge are taught on a wider scale and from an earlier age than before [35]. The results of a recent study [2] show that the use of Gen AI can significantly improve students' computational thinking skills, programming self-efficiency, and motivation. Gen AI allows for automating specific tasks such as coding and debugging, allowing students to focus more on higher-order thinking and

leading to a deeper understanding of programming concepts [2]. Additionally, in [39] Gen AI proves to be very accurate in explaining difficult code to students, as long as the code in question is correct. Meanwhile, M. Duong *et al.* [40] has shown that relying on AI tools alone may not suffice to boost student motivation when they encounter challenging programming tasks. Furthermore, code generated by Gen AI is not immune to LLMs' tendency to generate misleading or inaccurate information [26, 27] since research shows that Gen AIs' ability to solve programming problems at the level of higher education is quite inconsistent [11, 12, 41]. This variance in performance can prove dangerous for students lacking the necessary knowledge and skills to interact critically with it [28, 29]. Therefore, the potential of the technology to support programming education is high, but it is limited by its inconsistency in solving programming problems meant for higher education. Once these challenges are addressed, it is inevitable that the technology will find its place in higher education, much like past innovative technologies have been embraced.

2.2 Automatic assessment of tasks in programming higher education

Assessment of coding assignments in programming education is a task that grows more time-consuming with the number of students in a class [42]. A popular solution to address this challenge is to use automatic code-judging tools such as *Kattis* [21]. The general purpose of these tools is to act as a cloud-based, reliable, and impartial judge of programming code [43].

The evaluation process of automatic code-judging tools can be split into three phases: the submission phase, the judgment phase, and the feedback phase [43]. During the submission phase, the users input their code through the interface, and the automatic code judge compiles the code/ performs a static syntax check to judge whether the code can be run in the evaluation environment. The code is then tested against a predefined set of test cases during the judgement phase. For each test case, the tool checks whether the program ran without casting any errors, whether no predefined time and/or memory limits have been exceeded, and whether the program output matches what was expected. Lastly, binary results (i.e., pass/fail) on each test case are presented to the users during the feedback phase. In some cases, the

assignment creator can implement automatic grading in this phase based on time/memory consumption and/or the number of passed test cases [43].

2.2.1 Kattis

Various different automatic code-judging tools developed for different purposes exist [21, 44, 45]. Kattis [21] is one such tool; it is an automatic code-judge software developed mainly for educational practices in programming education. The main function of Kattis is to provide students with regular practice opportunities while offloading the responsibility of checking the correctness of student code submissions from the teachers to Kattis. This allows teachers and teaching assistants more time to teach their students. Furthermore, the benefit of using Kattis is a correct, impartial, and effective evaluation process [46].

Kattis is a web-based tool that allows programming educators to upload new assignments or use existing ones already on the platform in their courses. Kattis uses *black box* testing, which means that the evaluation process does not check the quality of the solution any further than if the submissions' output is correct. Besides universities, Kattis is also used as an automatic code judge in coding competitions and by companies in their hiring processes [21].

Result	**Description**
Accepted	The submission successfully passed the test case.
Compile Error	The submission did not compile correctly.
Wrong Answer	The submission output did not match the expected output of the test case.
Run Time Error	The submission threw an error during its run time, no further information is given.
Time Limit Exceeded	The submission took too long to solve the test case.
Memory Limit Exceeded	The submission used too much memory to solve the test case.

Table 2.1: Description of all possible results in Kattis [47]

For each assignment available in Kattis, a description of what the solution is meant to achieve is stated, along with a description of how the input and output of the solution are meant to be handled, as well as some example test cases [11]. The rest of the test cases are hidden from the user. All test cases of a solution must run successfully for the submission to pass the assignment. Therefore, Kattis tests each test case sequentially, and if a test case fails, the entire evaluation process stops, and the user is given feedback on what went wrong in the form of a predefined error message, with no further information on why the test case failed given to the user [47]. The six possible feedback

messages Kattis provides are tallied in Table 2.1.

Kattis also provides metadata around each assignment. This metadata contains information such as the maximum execution time each submission is allowed, the difficulty rating of the assignment, general statistics about the number of submissions, the number of accepted submissions, the number of successful users, and the relationships between these numbers. The difficulty rating of each assignment is a number between one and ten, where one is the easiest difficulty and ten is the hardest. Each assignment's difficulty rating is automatically decided based on the rate of successful submissions per submission [47].

2.2.2 AI-Generated Code Detection Tools

Another automation tool popularised by the emergence of more sophisticated and public LLMs, such as ChatGPT and Gemini, is tools to detect whether code has been generated by AI. This is referred to as AI-Generated Code Detection (AIGCD). Since the launch of ChatGPT, the tool has seen widespread use among students for their academic work [6]. This increased reliance on AI generated solutions significantly contributes to academic dishonesty and cheating [48]. This has led educators to utilize AIGCD-tools to establish whether students are cheating or not [49]. While existing tools have seen decent results in identifying AI-generated text [50], their effectiveness in recognizing AI-generated code is a different matter. Research into the subject shows that existing AIGCD-tools perform poorly when tasked with distinguishing between human and AI-generated code [51]. The unreliability of AIGCD-tools may result in disparities in the assessment of students' academic work, potentially leading to unfair grading.

2.3 Hallucination Mitigation

Due to the broad spectrum of subjects covered by modern LLMs, Gen AI tools have become vital sources of accurate and dependable information today [14]. In the academic setting, this statement holds very true [1, 6]. However, this increased reliance on Gen AI has sparked concern regarding LLMs' tendency to produce inaccurate or fictional information [22, 23]. This issue, commonly known as 'hallucinations', results in instances where advanced models such

as ChatGPT and Gemini may produce inaccurate references or facts that are fictional [14]. This issue presents a significant challenge since it can lead to the inaccurate acquisition of student knowledge, resulting in the tool hindering students' academic progress instead of supporting it.

Figuring out the reasons behind LLMs' hallucinations is difficult since the method LLMs derive their output through is a 'black box', making it unknown even to the engineers of the model. Nevertheless, there are some common causes for the issue. One such cause is the quality of training data. The data with which LLMs are trained can contain noise, errors, inconsistencies or biases [15, 16]. Furthermore, even if the data set is entirely reliable, the data may not cover every topic the LLM is expected to handle, leaving gaps for which the LLM has to fill by generalising its data without a method to verify its accuracy. Another possible cause of hallucinations in LLMs is the pattern generation techniques of the models, which can introduce their own biases and discrepancies [15].

Because of the increasingly integral role of Gen AI in society today, including educational practices, hallucinations can risk the spread of misinformation and affect learning processes and outcomes negatively. Therefore, the mitigation of hallucinations is a crucial area of concern in modern computational linguistics [16]. The widespread use of Gen AI in programming higher education [2, 6] makes LLM hallucinations a concern in that domain as well since it can communicate misinformation such as fictional libraries, unreliable code, deprecated functions and inefficient programming to the students. Besides early refinement in LLMs creation, researchers have presented a range of different strategies and techniques to tackle this issue. They include but are not limited to Few-Shot prompting and RAG [16, 19, 52], both of which have been proven effective in boosting the overall performances and accuracy of LLMs, as well as performing highly in several benchmarks [16, 17, 18].

2.3.1 Prompt engineering

Prompt engineering has emerged as a crucial technique for mitigating hallucinations and enhancing the capabilities of pre-trained LLMs [18]. It involves experimenting with various instructions to get the best output possible from a Gen AI model. Through carefully crafted instructions,

prompt engineering allows these models to excel in various tasks and fields [18, 53, 54]. The importance of prompt engineering becomes particularly clear when considering its influence in enhancing the adaptability of LLMs. This adaptability differs from conventional approaches, which require model retraining or extensive fine-tuning to achieve the same performance [18]. It extends the control to mitigate hallucination from the developers of LLMs' to the users, thereby making the techniques useful in settings such as higher education.

One example of prompt engineering is Few-shot prompting, which conditions a LLM to generate desired outputs by incorporating input-output examples, known as "shots", into a prompt [52]. Even providing just a few high-quality examples has been shown to improve LLMs' performances on complex tasks [18]. However, incorporating Few-shot prompting requires additional space within the prompt to accommodate these examples, which could be difficult for longer input texts due to limitations in the size of LLMs' context window as well as constraints on prompt lengths. Furthermore, while the selection and structure of "shots" can significantly influence model behaviour, pre-trained biases and hallucinations may still affect the output [55]. Therefore, while few-shot prompting enhances capabilities for complex tasks such as programming tasks [12], careful prompt engineering is critical to mitigating unintended model biases and hallucinations.

2.3.1.1 Retrieval Augmented Generation

Retrieval Augmented Generation (RAG) is another mitigation strategy that improves the performance of LLMs by accessing external, reliable knowledge instead of relying solely on potentially outdated, unreliable training data [16]. This approach directly tackles the significant challenge of maintaining accuracy and relevance in LLM output. RAG has been shown to effectively address the issue of generating misleading or erroneous information (hallucinations) by producing responses that are not only relevant and up-to-date but also verifiable [16, 17].

The RAG system operates by taking an input prompt and fetching a collection of relevant data from a specified source. These data are then combined into a cohesive context alongside the initial input prompt and are subsequently provided to the LLM, which generates the final output. This

adaptive approach proves invaluable in scenarios where factual information may change over time, a capability not inherent in static knowledge bases typical of LLMs. RAG circumvents the need for extensive retraining, instead facilitating access to the most current information for generating dependable outputs through retrieval-based generation [56].

Considerable research efforts have been performed on the performance of RAG in generating text, where it has scored highly on several benchmarks, such as Natural Question [57], WebQuestions [58], and CuratedTrec [19, 59]. This shows the potential of RAG as an option for enhancing the reliability of LLM output. However, to our knowledge, little research has been performed on using RAG techniques for mitigating hallucination in LLM attempts at generating programming code.

2.4 Related work

This section provides an overview of related work where Gen AI technologies have been used, tested and improved in the areas of programming higher education and hallucination mitigation. This section begins by showing research results linked to how well generative AI can solve coding tasks in higher education. This is followed by showing how different prompt optimisation methods have performed in the text generation field. Lastly, a presentation shows how RAG (that is central to the present work) has been used to improve AI code generation before.

2.4.1 AI and code generation

With the launch of ChatGPT, Gen AI tools have gained increased focus in research and development [1]. The topic of how well these tools are at solving tasks of different kinds has been of particular concern, with their ability to solve programming tasks being the focus of this section. Research shows mixed results when tasking LLMs to solve programming tasks. It all depends on the model being tested, the prompt engineering techniques applied as well as the nature of the programming task at hand. But in general, research has demonstrated that commercial LLMs such as ChatGPT and Gemini perform well on simpler programming tasks but struggle with solving more difficult

ones [11, 12, 60, 61].

Dunder *et al.* examined ChatGPT's potential impact on academic integrity [11]. An experiment was undertaken to evaluate ChatGPT's ability to generate code solutions for introductory programming course assignments sourced from Kattis [21]. 127 assignments of evenly distributed difficulty levels were chosen for the experiment. Each test was performed by using an assignment description as a prompt for ChatGPT, giving the LLM one try at solving the task. The generated code was then given to Kattis for automatic assessment. Their findings revealed that ChatGPT independently solved 19 out of 127 programming tasks. The 19 assignments that were solved all belonged to the lower difficulty levels on Kattis, illustrating ChatGPTs' proficiency in generating accurate code solutions for simple problems on Kattis. Yet, it struggled with more complex programming tasks. Additionally, their findings show that ChatGPT managed to partly solve more complex assignments, meaning that it failed on certain but not all test cases for the assignments on Kattis. This highlighted the potential of the technology to solve more difficult programming assignments in the future.

Similarly, Geng *et al.* explored how well ChatGPT can perform in an introductory-level programming course [12]. In their work, they treated ChatGPT as a student in a real functional language programming course at an anonymous university. The course introduced its students to functional programming and programming paradigms and had more than 300 undergraduate students registered. The course consisted of 10 programming assignments, a midterm and a final exam, all of which were given to ChatGPT to solve. The testing was split into two phases: the unassisted phase and the assisted phase. In the unassisted phase, assignment descriptions and requirements were fed into ChatGPT as prompts, and the resulting code generation was evaluated as is. However, various strategies were implemented in the assisted phase to improve the LLMs' response. These strategies included paraphrasing input prompts, providing natural language hints, teaching by example and providing test cases. The results of both phases show that ChatGPT could pass the class. The LLM ranked 220th out of 314 students in the unassisted experiment and 155th out of 314 students when being assisted.

Extensive research has explored the capabilities of LLMs in code generation. Various LLMs, ranging from general-purpose models like ChatGPT, Gemini, and their predecessors to specialized models tailored for

programming tasks such as AlphaCode [62], Copilot [63], and Codex [64], have undergone thorough testing. The outcomes of this research present a mixed picture, with numerous studies praising LLMs for their ability to produce high-quality code across diverse programming tasks with a low error rate [12, 65, 66, 67], while other papers report large inconsistencies in results depending on differences in prompt structures or model types. These inconsistencies often lead to unsuccessful attempts along with, at times, incoherent code when LLMs are tasked with code generation [11, 60, 61]. However, what is prevalent in all of these papers is the inconsistency modern LLMs showcase when solving programming tasks of various difficulties. Regardless of the task, the LLMs always generates code that claims to solve the assignment, although this does not always turn out to be true. This highlights the prevalence of hallucination in modern LLMs, which needs to be mitigated to achieve the full potential of the technology.

2.4.2 Hallucination mitigation techniques

Several hallucination mitigation techniques have been researched in recent years. These techniques include model retraining, fine-tuning or prompt engineering to achieve an improved text generation performance [18].

Tonmoy *et al.* examined over thirty techniques developed to mitigate hallucination in LLMs [16]. Furthermore, they categorised these methods based on various parameters to distinguish the different methods that tackle the hallucination issue. Firstly, they distinguish between hallucination mitigation techniques that can be used on already trained models and techniques that involve the development of new models. The techniques that can be used on already trained models are mainly various forms of prompt engineering strategies. This includes different implementations of RAG, Self-refinement Through Feedback, and Prompt Tuning [16]. The premise of Self-refinement Through Feedback techniques is that the output of an LLM can be improved and made more accurate over multiple prompt iterations by providing proper feedback about the output to the LLM at the end of each iteration [68]. Prompt tuning refers to the process of refining the instructions given to a pre-trained LLM during the fine-tuning stage, aiming to enhance the model's performance for specific tasks [69]. The paper provides a valuable resource for researchers aiming to gain a thorough understanding of the present state of hallucination in LLMs and the approaches developed to tackle this urgent issue [16].

While much research has been performed to mitigate hallucinations in text generation tasks, the same techniques have not been explored as extensively in the domain of programming code generation. This is a concern since hallucinations in code generation can communicate misinformation such as fictional libraries, unreliable code, bad practices, deprecated functions and inefficient programming to the users.

2.4.3 Iterative RAG for code generation

The RAG technique has proven highly effective in mitigating hallucination in text generation tasks [16], but these results do not necessarily translate to programming tasks. Despite the similar challenge of maintaining coherence and consistency in the input context, the unique syntax and structure of programming assignments present distinct hurdles for the RAG approach.

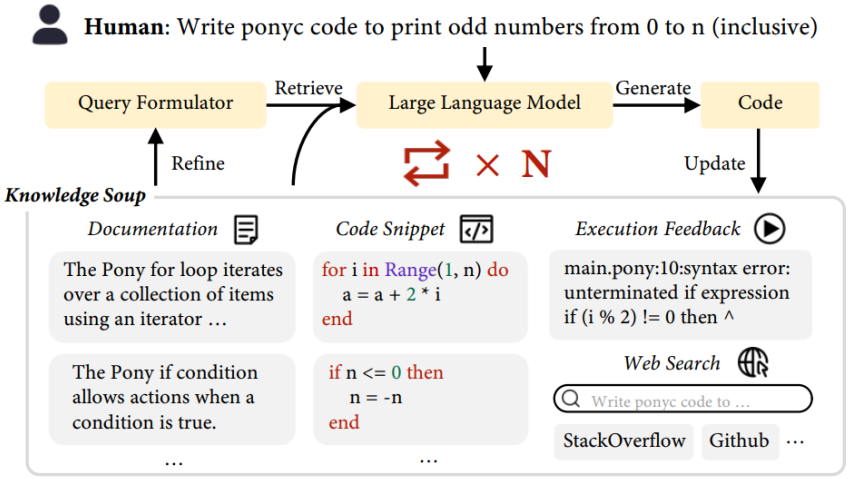

Figure 2.1: The ARKS process [70]

Su *et al.* [70] have explored the effects of using RAG to improve the code generation of the LLMs, namely ChatGPT and CodeLlama. Their work developed an Active Retrieval in Knowledge Soup (ARKS) method to accomplish this goal. ARKS is a prompt optimisation method that learns from and extends the RAG method by supplying the LLM with further context to their prompt (see Figure 2.1). Unlike the classical RAG, which usually uses

a prompt combined with a static knowledge base to generate a final response in a single round, ARKS continues the process by actively refining queries and updating the knowledge base, which is known as the 'knowledge soup' in ARKS. The 'knowledge soup' of ARKS integrates multiple diverse sources of information to enhance the code generation of the LLMs. These sources are:

- Web Searches
  By extracting keywords from the previous prompt and knowledge soup, a Google search is performed. ARKS then retrieves the top-ranking results, which are converted into markdown text and incorporated into the knowledge soup.

- Documentation
  ARKS includes documentation related to the programming language, libraries and frameworks that are used in the code into the knowledge soup.

- Execution Feedback
  Feedback obtained from executing generated code solutions is incorporated into the knowledge soup, enabling the model to learn from its own output.

- Generated code
  Previous code snippets that are generated by the model are included in the knowledge soup.

Incorporating the 'knowledge soup' led to a substantial increase in the average accuracy of generated code for ChatGPT and CodeLlama, by 21.2% and 17.4%, respectively. This illustrates the importance of carefully refining queries and actively updating knowledge in an iterative retrieval process for LLMs to achieve superior effectiveness in generating code.

Based on the survey of previous work, it is decided that the present thesis will investigate a RAG based approach to mitigate hallucinations in code generation from modern LLMs. Additionally, the scope of the experiments will be limited to code generation for higher education programming.

Chapter 3

Method

This chapter presents the methods used in the thesis to accomplish its purpose. The research process is presented first, followed by multiple sections on the chosen experiment parameters and their justification. Lastly, this chapter discusses the study procedure along with the evaluation process.

3.1 Research Process

To answer the research questions posed in section 1.2, the four sub-goals from section 1.3 need to be fulfilled:

1. A significant number of programming assignments of various difficulties need to be chosen.
2. The capabilities of ChatGPT and Gemini must be examined by generating code for the programming assignments without the use of RAG or any other prompt engineering techniques.
3. RAGs' capabilities need to be examined by optimising the input prompts for ChatGPT and Gemini when generating code for the programming assignment.
4. The results of both runs need to be compared and analysed.

To plan the experiment correctly, research was conducted into the state of Gen AI in higher education today, what LLMs were most relevant for this experiment, what prompt engineering techniques exist and their limitations, as well as how to best assess the results of the experiments. Secondly, the

experiment was conducted. This was accomplished by testing the capabilities of the two selected LLMs, namely ChatGPT and Gemini, in completing programming assignments meant for higher education, with and without using prompt engineering techniques to improve the results.

3.2 Research Paradigm

The chosen method for the thesis was an experimental case study, selected to delve deep into RAGs ability to mitigate hallucinations in Gen AI within the context of programming higher education. This experimental case study design enabled us to explore the nuances of the problem within a controlled yet realistic setting, resulting in meaningful conclusions for the chosen field.

3.3 LLMs

The LLMs that were chosen for the experiment were ChatGPT-3.5 and Gemini Pro. The choice was mainly based on the LLMs' popularity. The models have amassed approximately one hundred million and eighty-three million weekly users, respectively, as of February 2024 [3, 4]. Unlike various models that are created and fine-tuned for the specific task of solving programming tasks, ChatGPT and Gemini are general-purpose models, but despite this, the models perform on a level with most state-of-the-art LLMs in every traditional benchmark designed for machine learning while still scoring high in programming benchmarks [71, 72]. See Table 3.1.

Benchmark	Description	Gemini Pro	GPT-3.5	Best external LLM
MMLU [73]	Multiple-choice questions in 57 subjects (professional & academic)	79.1%	70.0%	79.6%
HellaSwag [74]	Commonsense reasoning around everyday events	84.7%	85.5%	89.0%
MATH [75]	Math problems across 5 difficulty levels & 7 subdisciplines	32.6%	34.1%	34.8%
Natural2Code [76]	Python code generation	69.6%	62.3%	-
HumanEval [77]	Python coding tasks	67.7%	48.1%	70.0%
DROP [78]	Reading comprehension & arithmetic	74.1%	64.1%	82.0%

Table 3.1: Gemini Pro & ChatGPT-3.5 benchmark results compared to state of the art, excluding Gemini Ultra & ChatGPT-4 (as of December 2023)[72, 79]

3.4 Programming assignments and their assessment

As discussed in chapter 2.2.1, Kattis is an automatic code-judging tool specifically developed for use in higher education; the assignments uploaded to Kattis are often used in programming higher education. This made Kattis highly suitable for this paper's experiments.

The examined coding tasks were randomly selected from Kattis based on only two requirements. Firstly, the assignment description on Kattis had to easily be able to be converted into a LLM prompt. That is, the assignments with a large number of mathematical equations, input files and figures were excluded due to technical constraints. Secondly, examples of solutions for the assignment had to be freely available online. These were later used in the prompt engineering through RAG.

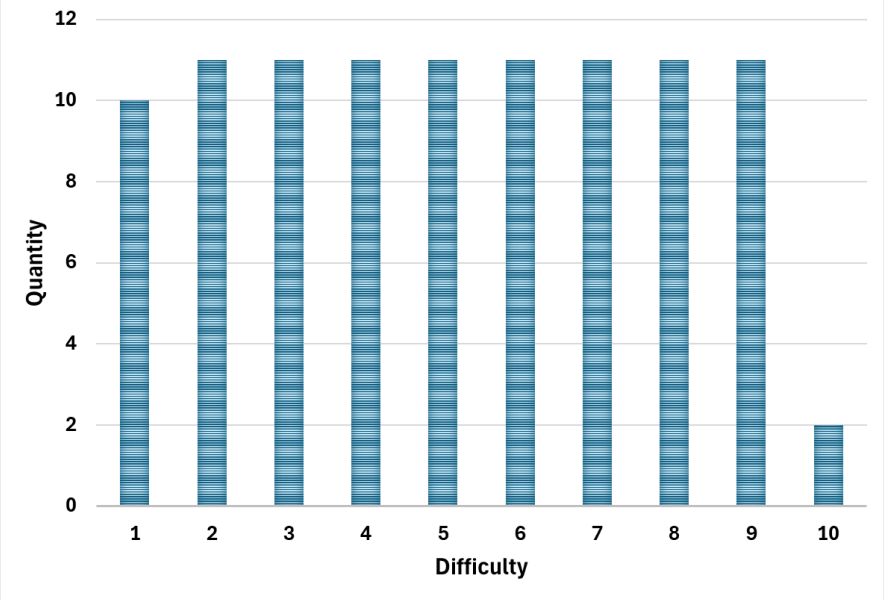

Figure 3.1: Distribution of the 100 assignments from Kattis based on difficulty.

100 assignments were chosen. Each assignment in Kattis comes with a difficulty rating ranging from 1 to 10, where 1 is the easiest difficulty and 10 is the hardest. The difficulty rating of each assignment is automatically decided based on the rate of successful submissions per submission [47]. All the

assignments selected for the experiments were evenly distributed in difficulty, except for the highest difficulty, 10, as can be seen in Figure 3.1. The reason for this disparity was that there were not any more assignments of difficulty 10 that fit the requirements described earlier available on Kattis [47].

Besides the difficulty rating, Kattis also provides general statistics about the number of submissions, the number of accepted submissions, the number of successful users, and the relationships between these numbers. These statistics were used to calculate the average number of attempts it takes to solve the assignment. Information about all the assignments and their statistics can be found in Appendix A.

3.5 Study Procedure

As previously mentioned in section 3.1, the experiment was split into two test rounds. The first test round aimed to find the base capabilities of ChatGPT and Gemini, while the second test round aimed to find the hallucination mitigation capabilities of RAG. In both test rounds ChatGPT and Gemini attempted to solve the 100 assignments in as few attempts as possible. Since each output from a Gen AI-tool claims to solve the problem posed to it in the prompt, any output that failed in that endeavour was considered a hallucination by the LLM. The number of attempts the LLMs was given for each assignment was based on the average number of attempts per user the assignment had on Kattis but was capped at five attempts to ensure the prompts stayed within the models' context window. The main difference between each round was in the structure of the initial prompt the LLMs got, as well as in the feedback that was provided to the LLMs in each attempt following the first.

3.5.1 Prompt Structure and Limitations

Since the tests were performed on both ChatGPT-3.5 and Gemini-Pro, certain limitations had to be considered for the prompts to work seamlessly on both models. Firstly, the prompts had to be converted to pure text since ChatGPT-3.5 does not have the capability to process other types of input [71]. Secondly, the combined prompt length for all attempts had to be below sixty thousand characters in total since that is the approximate size of the ChatGPT-3.5 context

window, which is smaller than the context window of Gemini Pro [71, 72].

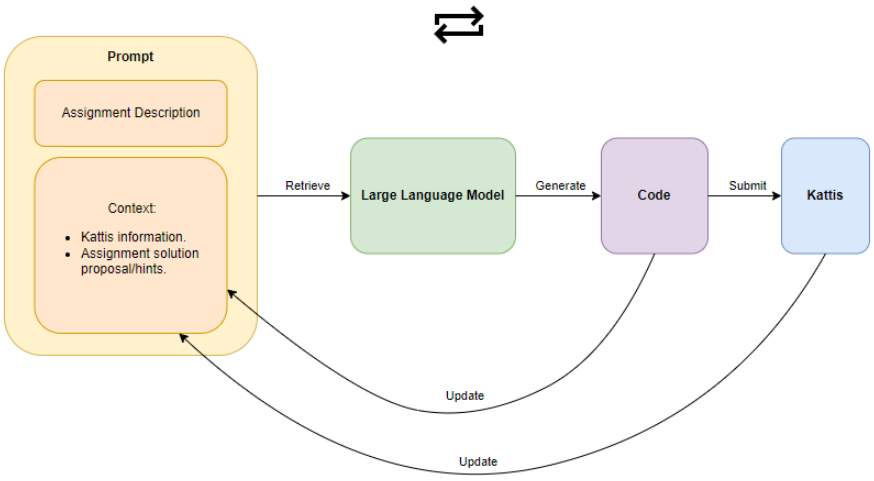

Figure 3.2: The RAG test procedure. This repeats for however many attempts an assignment gets or until the code submitted is accepted.

For the first test round, the prompt for each assignment consisted only of Kattis's assignment description. When a failed attempt occurred, no further feedback except the previous code solution and the fact that the code failed was given to the LLMs before their next attempts. For the second test round, RAG was utilized in each prompt. This meant that besides the assignment description, the prompt would contain further context in an attempt to improve the performance of the LLMs, see Figure 3.2. The context consisted of information about the Kattis platform, previously generated code, Kattis feedback from the previous attempts, and lastly, solution proposals/hints to the assignment found online. The information about the Kattis platform provided to the LLMs was acquired from [47] and mainly revolved around how Kattis handles input/output, how the solutions are compiled and run on Kattis, the possible feedback Kattis provides and what they mean, and the Python modules Kattis allows in its solutions. The solution proposals/hints were acquired from working solutions for the assignments found through a simple Google search. No actual code was provided to the LLMs. Instead, broad ideas and techniques, like, for example, ”The problem can be solved using Djikstras algorithm along with a pairwise comparison”, ”The elements can be tracked using a Disjoint Set data structure”, or ”Prime factorization can be utilized to solve parts of the problem”, were extracted from the solutions without

detailing how the code actually works. When a failed attempt occurred, one of the five feedback messages found in Table 2.1 was provided to the LLM along with the previous code solution and the number of test cases that were passed. Kattis provides a small amount of test cases for each assignment but keeps the rest hidden. Therefore, in the case that an assignment failed on one of the known test cases, the input and expected output of the test case were also provided as feedback to the LLMs. In case a compilation or run time error occurs, the error message provided by Kattis, if provided at all, was also part of the feedback.

3.6 Evaluation framework

The results of the experiments were evaluated using a quantitative method on the descriptive statistics acquired from the testing. The performance of both methods was assessed by comparing the percentage of problems successfully solved to the total number of problems. Additionally, the number of unsuccessful attempts for each method was analyzed alongside the frequency of each error message generated by Kattis. Furthermore, analysis was performed on the highest-rated difficulty each method succeeded in solving. Moreover, a breakdown of each method's ability to improve its output over multiple attempts is performed. Lastly, a detailed analysis was carried out on the performance of each individual LLM based on the employed method.

Chapter 4

Results and Analysis

The results presented in this chapter are from the 400 tests done in order to examine the effectiveness of RAG in mitigating hallucinations by Gen AI within the context of programming higher education, as described in section 3.5. First, the overall success rate between using RAG and not using it is presented in section 4.1. Followed by a further breakdown of the frequency of each error message generated by Kattis in section 4.2. Finally, the ability of each method to improve over time alongside each method's highest-rated difficulty successfully solved is analysed.

4.1 Success rate

In total, ChatGPT and Gemini attempted to solve a total of 100 assignments each from Kattis. These attempts were made both with and without the use of RAG, conducting 400 tests in all, with up to five attempts for each test. Figure 4.1 illustrates the outcomes of these tests. When employing RAG for prompt engineering, the LLMs achieved success in 24% of the assignments, as opposed to only 14% without RAG. This shows a statistically significant performance improvement when utilizing RAG over basic prompts for code generation tasks. Specifically, RAG resulted in a 10% increase in the acceptance rate on Kattis and a notable 13% decrease in the failure rate. This suggests that using RAG led to 13% more assignments being at least partially completed instead of completely failing them. When tested for their statistical significance, these results resulted in a p-value of 0.001132. A low p-value, such as 0.001132, indicates that the observed results are unlikely to have occurred by chance. In simple terms, it suggests that there is strong

evidence that the effect of RAG observed in the study is probably real and not just due to random variation.

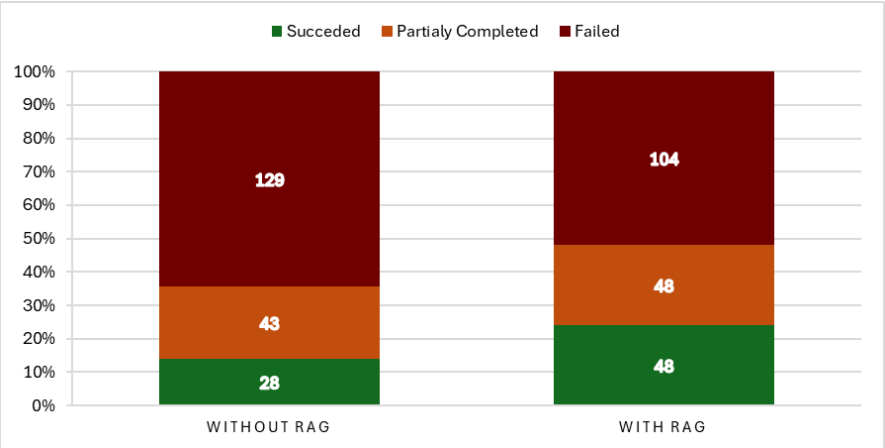

Figure 4.1: Overall results of generating code for the 100 assignments when using RAG and not using RAG.

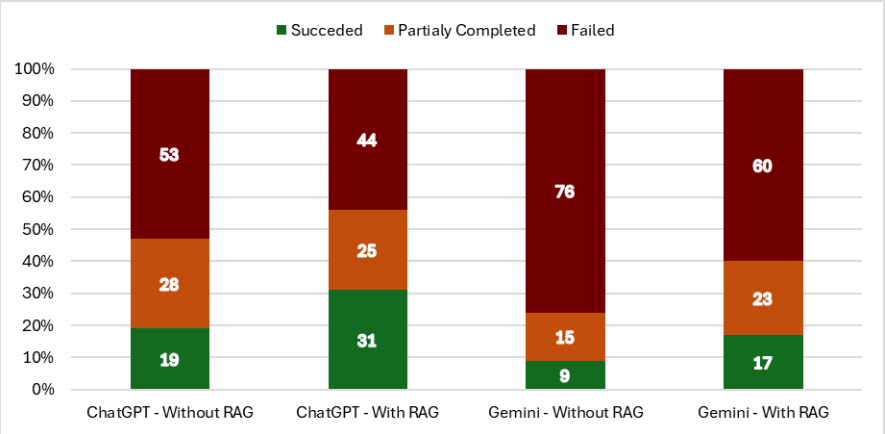

Figure 4.2: Further breakdown of the overall results per LLM and prompt engineering method respectively.

Figure 4.2 provides a detailed breakdown of the experiment's overall results into its four main components. Utilizing RAG enhanced the performance of both LLMs. However, regardless of the prompt engineering method, ChatGPT significantly outperforms Gemini in solving the code

assignments. These findings were unexpected, considering that Gemini-Pro performed notably better than ChatGPT-3.5 in the major Python coding benchmarks, as illustrated in Table 3.1.

4.2 Kattis' response distribution

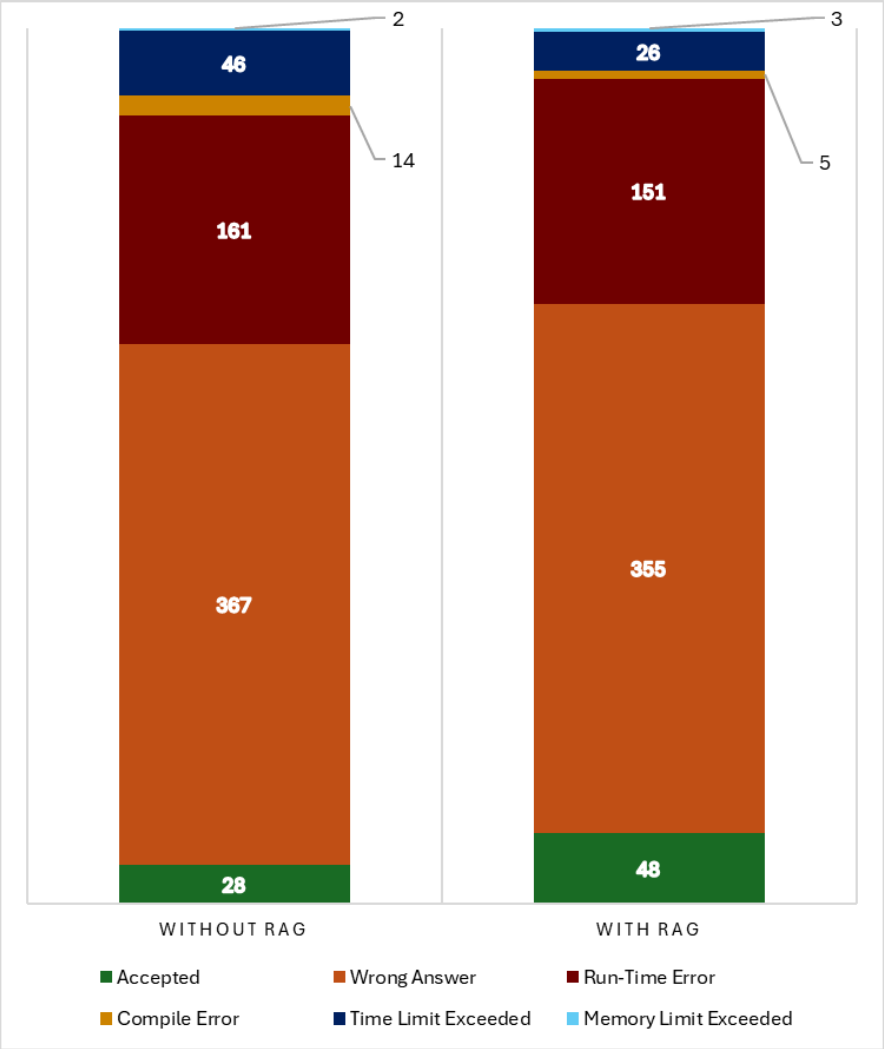

Figure 4.3: Breakdown of Kattis's responses per submission, both with and without utilizing RAG.

In section 2.2.1, it is detailed that Kattis can return six predefined error

messages upon submission of a solution. These messages are outlined in Table 2.1. Over the course of the study, a total of 1207 attempts were submitted to Kattis. The distribution of Kattis's responses for each submission is illustrated in Figure 4.3. In the case of failure, regardless of whether RAG was utilized or not, the most common response was *"Wrong Answer"*, occurring 60.37% (with RAG) and 59.39% (without RAG), respectively. The second most frequent response in such cases was *"Run-Time Error"*, observed in 25.68% (with RAG) and 26.05% (without RAG) of attempts. These findings suggest that while RAG-assisted code often exhibits unexpected behaviour in terms of its logic, it also generates faulty code less frequently compared to non-RAG-assisted code.

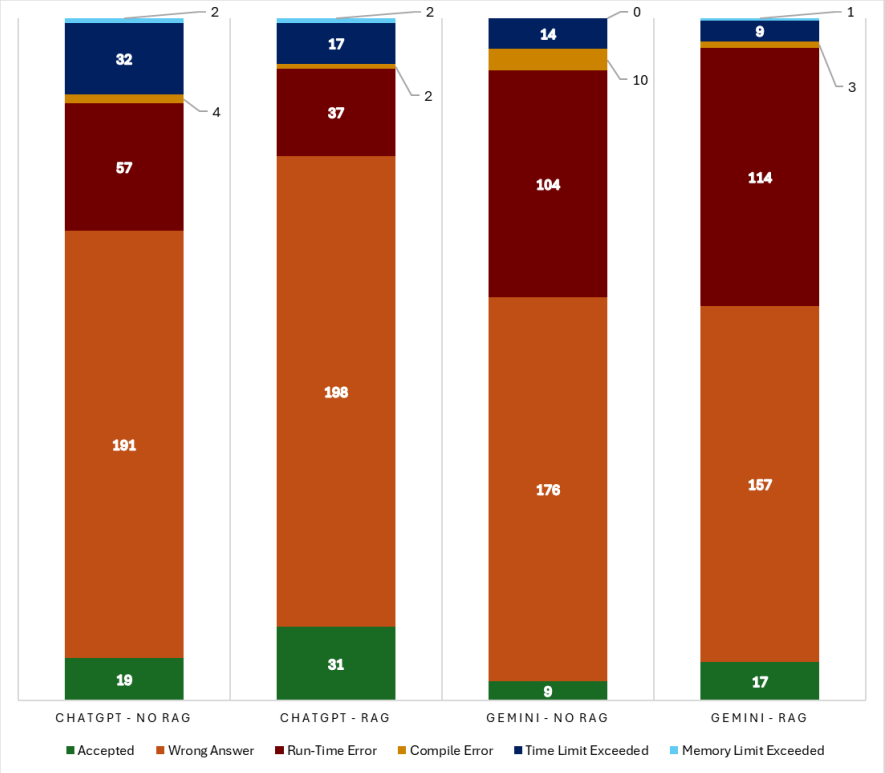

Figure 4.4: Distribution of Kattis responses for each submission separated based on LLM and usage of RAG.

The distribution of Kattis's responses for each submission is broken down into responses according to the LLM and prompt engineering method utilized in Figure 4.4. Although *"Wrong Answer"* remains the predominant

response across all test groups, there is a notable increase in *"Run-Time Error"* responses for the Gemini tests.

4.3 Improvement through feedback

Code improvement over time

Method	ChatGPT	Gemini	Average
No RAG	10.47%	8.45%	9.46%
RAG	18.95%	10.95%	14.95%

Table 4.1: Percentage of attempts that improve on their previous attempts.

One significant difference between the prompt engineering methods tested in the study pertained to the absence of feedback provided to the LLMs after each failed attempt in the case where RAG was not utilized. This resulted in a drastic difference in the rate Kattis's responses improved on the previous attempt, as can be seen in Table 4.1. In the study, an improvement in Kattis's response occurs in three different ways. Firstly, an unsuccessful attempt followed by a successful one is considered an improvement. Secondly, any increase in the number of successful test cases, regardless of the error message received from Kattis, is deemed an improvement. Lastly, if an attempt yields an equal number of successful test cases but Kattis's response shifts from either *"Run-Time Error"* or *"Compile Error"* to *"Wrong Answer"*, *"Time Limit Exceeded"*, or *"Memory Limit Exceeded"*, it counts as an improvement. Table 4.1 illustrates how the utilization of RAG led to a 5.49% rise in output improvement over multiple attempts for the LLMs. Another notable finding from Table 4.1 is that, regardless of RAG being used or not, ChatGPT outperforms Gemini in output improvement over multiple attempts.

4.4 Assignment difficulty

Figure 4.5 displays the difficulty rating of each assignment that the LLMs attempted during the experiments. The most difficult assignment successfully solved utilizing RAG had a rating of 5.9, compared to a rating of 2.7 when RAG was not utilized. This suggests that RAG has a significant impact on ChatGPT

and Gemini's abilities to solve coding assignments of a higher difficulty. There were no major differences between the LLMs regarding assignments with a high difficulty. Both ChatGPT and Gemini solved the assignment of difficulty 5.9 when utilizing RAG.

The least difficult assignment failed with RAG had a rating of 2.1 compared to a rating of 1.3 without RAG. The difference in consistency of each method in solving assignments of a lower difficulty is also notable. When focusing solely on the third of the assignments with the lowest difficulty rating, the success rate increases to 79.41% when utilizing RAG compared to 55.88% without RAG. Furthermore, the difference is even more apparent depending on the LLM being used. Gemini struggled drastically more than ChatGPT when it came to simpler assignments, regardless of whether RAG was employed or not. When focusing solely on the third of the assignments with the lowest difficulty rating, the success rate of Gemini was lower than for ChatGPT by 32.35% when utilizing RAG and 29.41% when RAG was not utilized. Detailed test outcomes are available in Appendix B.

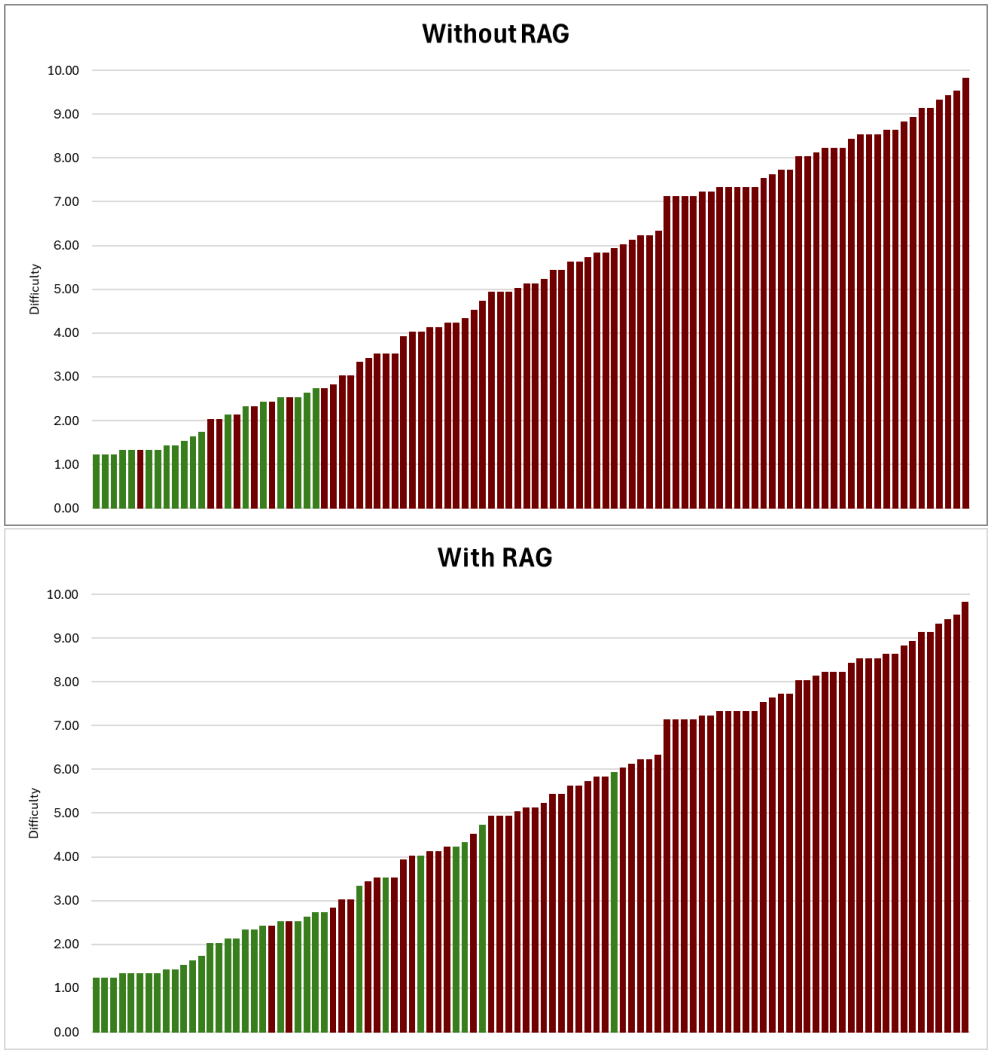

Figure 4.5: Difficulty distribution of assignments involved in the experiments, green bars were accepted by Kattis, while red bars were not.

Chapter 5

Discussion and Future work

This research aimed to investigate how effective the RAG technique is in reducing hallucinations generated by Gen AI within the settings of programming higher education. This chapter begins by exploring how the findings from the preceding chapter address the two research questions:

- **RQ1:** To what extent can ChatGPT and Gemini accurately solve programming assignments in the setting of higher education?
- **RQ2:** To what extent can prompt engineering through RAG mitigate hallucination in ChatGPT and Gemini code generation?

The chapter then delves into a section outlining the contributions of this thesis to the practical field. Finally, the chapter discusses the limitations of this study and suggestions for future improvements.

5.1 Assessment of Research Questions Outcomes

The findings outlined in Figure 4.2 provide the most comprehensive response to **RQ1**. The extent to which ChatGPT and Gemini can accurately solve programming assignments in the setting of higher education varies significantly, influenced not only by the model utilized but also by the approach to prompt engineering. Regardless of the prompt engineering method, ChatGPT consistently outperformed Gemini in accurately solving programming assignments, reinforcing earlier observations in [61]. Regarding the effectiveness of ChatGPT and Gemini in solving programming

assignments within higher education, the results indicate that both models consistently succeed in handling simple problems regardless of the prompt engineering method utilized. However, they struggled with more complex programming tasks, aligning with previous findings in [11, 12].

On the other hand, the results depicted in Figure 4.1 offer a comprehensive response to **RQ2**. They demonstrate a significant improvement in the performances of ChatGPT and Gemini when employing RAG, which aligns with the findings of [16, 70]. Furthermore, the utilization of RAG not only reduces the error rate of generated solutions but also enhances the LLMs' capability to improve on their previous attempts, thus further supporting the conclusions drawn in [70]. Since each output from a Gen AI-tool claims to solve the problem posed to it in the prompt, any output that fails in that endeavour is considered a hallucination by the LLM. With this perspective, Chapter 4 effectively addresses **RQ2**.

5.2 Contributions to research and the field of practice

The findings presented in this thesis offer novel insights into the capabilities of RAG to address hallucinations produced by ChatGPT-3.5 and Gemini Pro in programming education. Section 4.1 demonstrates results showing a significant improvement in the performance of Gen AI through the integration of RAG, irrespective of the underlying LLM utilized. Specifically, the results outlined in Section 4.2 indicate that employing RAG results in a reduced error rate coupled with increased acceptance rates. These observations align with prior research [16, 70], bolstering the reliability of RAG across different contexts, including programming higher education. Considering the potential synergy with findings from [65, 66, 67] is also interesting, imagining the possible performance improvement achievable by combining RAG with advanced models tailored for programming tasks like AlphaCode [62], Copilot [63], and Codex [64].

The findings of this thesis shed light on the coding capabilities of both ChatGPT and Gemini. While both models exhibit a consistent ability to solve simpler code tasks, they struggle as the difficulty of assignments increases, aligning with prior research [11]. Notably, both models also improved their performance when given proper feedback and multiple attempts at

a problem, which confirms the findings in [12]. Surprisingly, ChatGPT-3.5 notably surpasses Gemini Pro across all metrics outlined in Chapter 4, contradicting earlier claims in [72]. Additionally, an intriguing observation during experimentation was Gemini's tendency to exhibit a unique form of hallucination. On multiple occasions, the model adamantly refused to generate alternative code attempts, asserting the correctness of its previous solution and challenging the assessment by Kattis. Although this was not investigated further than being noted during the experiment, it underscores a distinctive challenge specific to Gemini.

The results of this thesis, alongside mounting evidence of Gen AIs' propensity to solve coding assignments in the setting of higher education [11, 12], raise questions about academic integrity. These questions become a serious concern when considering the ineffectiveness of AIGCD-tools, as elaborated in section 2.2.2. In introductory programming courses, such as the one referenced in [12], educators must consider the possibility of students using Gen AI to pass the coursework. Therefore, preemptive measures must be considered, such as changing how these courses are graded. Alternatively, a more proactive approach could involve encouraging the usage of Gen AI in these courses, perhaps even incorporating a segment on effective tool utilization. Such an approach would anticipate the evolving landscape of professional environments, where LLMs are likely to play an increasingly prevalent role as their accuracy and performance improve over time.

Even in the present day, simpler programming tasks can likely be automated using Gen AI within professional settings, as evidenced by the findings of this thesis. Furthermore, under the guidance of seasoned developers, Gen AI could take on more complex tasks. This is evident from the outcomes highlighted in Section 4.1, where ChatGPT and Gemini managed to generate solutions that passed some test cases for many of the tougher assignments on Kattis. Therefore, with an experienced developer utilizing the tool, coding can be automated extensively, requiring only minimal debugging from developers in instances where Gen AI does not manage to solve the task completely. We already see this type of usage for LLMs today in tools such as Copilot [63].

5.3 Limitations

This section delves into potential limitations that may have impacted the outcomes of the study. One such limitation concerns the assignment hints provided through the RAG prompting. Since there are multiple approaches to solving a coding problem, basing these hints on just one solution, which may not be the optimal one, could introduce a bias in the results. One possible improvement could have been to explore multiple solutions, but this would have required a significant amount of time that was not available.

Another potential limitation relates to the structure of the prompts used during the experiment. As discussed earlier, Gemini demonstrated some peculiar behaviour that ChatGPT did not exhibit. Specifically, Gemini occasionally declined to generate new code attempts because it believed Kattis was making an error in its judgment. However, the reasons behind Gemini's behaviour were not investigated due to time constraints and this being outside the scope of the thesis. It is possible that the prompt structure favoured ChatGPT, thereby potentially skewing the results in its favour. An improvement could involve conducting further research into optimal prompting practices for Gen AI, as well as conducting general testing to determine the prompt structure that brings out the best of both models before conducting the experiment.

Lastly, the sample size for the experiment could have been larger, both in terms of the number of assignments used and the number of models tested. While the samples utilized in this case study are sufficient to draw conclusions from the results, a larger sample size could uncover additional insights.

5.4 Sustainability and Ethical Implications

Extensive research has been conducted on the environmental impact of training and using LLMs like ChatGPT and Gemini [80, 81]. Studies indicate that LLMs produce substantial carbon dioxide emissions, primarily due to the data centers required to maintain the models [80]. Additionally, they have a significant freshwater footprint necessary for model training [81]. Although this thesis did not directly address these environmental issues, its findings have the potential to mitigate them. Utilizing RAG provides an alternative to

retraining models when they fail specific tasks. The thesis demonstrates that RAG supplies models with the flexibility to handle complex, domain-specific tasks, such as coding, through prompt engineering rather than retraining or extensive fine-tuning. This adaptability can reduce both carbon dioxide emissions and the freshwater footprint associated with training new models.

The thesis results raise significant ethical concerns about academic integrity. The findings demonstrate that using RAG enables ChatGPT and Gemini to, at least partially, complete nearly 50% of the assignments used in the thesis experiments. Students could exploit these partial solutions by making minor adjustments and submitting them, thereby earning grades that do not reflect their true programming abilities. This issue is one that educators and universities must consider moving forward, especially since LLMs, like ChatGPT and Gemini, are continually improving. Unintentionally, the thesis provides a detailed guide on how to leverage ChatGPT and Gemini with RAG to cheat on higher education programming assignments on Kattis. However, it also highlights the threat modern LLMs pose to academic integrity and offers valuable insights for further research into combating these challenges. It is evident that modern LLMs cannot be overlooked in curriculum planning for higher education programming; both teaching methods and grading systems need to be adapted accordingly.

5.5 Future work

There are several potential directions for advancing this research in the future. One avenue involves enhancing the RAG method utilized in this thesis by broadening the range of sources used to contextualize prompts. This expansion could include top internet search results, course material relevant to the assignment, previously generated solutions from the model, and programming documentation.

Another path for extending this research involves automating the RAG process through the development of a new Gen AI-based tool, leveraging existing models like GPT-3. This tool could automate the creation of RAG contexts for prompts related to coding problems, exploring if the results of this thesis can be replicated through this automated tool.

Additionally, further exploration could focus on identifying other prompt

engineering techniques that can be adapted for code generation apart from RAG or, alternatively, alongside RAG.

Moreover, one can apply the method outlined in this thesis to different contexts beyond higher education, such as coding competitions and hiring processes. This could offer valuable insights into the performance of RAGs in other real-world scenarios.

These suggestions represent just a few potential avenues for advancing this research.

Chapter 6

Conclusions

This paper explored the RAG techniques' ability to mitigate hallucination in the setting of programming higher education. The findings indicate a significant performance improvement for both models examined when incorporating RAG. This underscores the versatility and effectiveness of the RAG approach, as it demonstrates improved performances for code and text generation. Additionally, the findings suggest a need for changes within the higher education setting to address the new avenues of learning and potential cheating facilitated by modern Gen AI technologies. Preemptive measures must be taken to accurately evaluate students who may have leveraged Gen AI in their coursework. Moreover, it is essential to incorporate prompt engineering techniques into the curriculum to empower users of Gen AI to utilize these tools effectively while minimizing the risk of generating hallucinations. This approach would prepare students for the professional environment of the future, wherein Gen AIs are likely to assume a more prominent role as their accuracy and performance continue to advance.

References

[1] T. Nazaretsky, M. Cukurova, and G. Alexandron, "An instrument for measuring teachers' trust in ai-based educational technology," in *LAK22: 12th International Learning Analytics and Knowledge Conference*, ser. LAK22. New York, NY, USA: Association for Computing Machinery, 2022. doi: 10.1145/3506860.3506866. ISBN 9781450395731 p. 56–66. [Online]. Available: https://doi.org/10.1145/3506860.3506866 [Pages 1, 10, and 13.]

[2] R. Yilmaz and F. G. Karaoglan Yilmaz, "The effect of generative artificial intelligence (ai)-based tool use on students' computational thinking skills, programming self-efficacy and motivation," *Computers and Education: Artificial Intelligence*, vol. 4, p. 100147, 2023. doi: https://doi.org/10.1016/j.caeai.2023.100147. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2666920X23000267 [Pages 1, 7, 8, and 11.]

[3] F. Duarte. (2024) Number of chatgpt users. [Online]. Available: https://explodingtopics.com/blog/chatgpt-users [Pages 1 and 20.]

[4] R. Shewale. (2024) Google gemini statistics. [Online]. Available: https://www.demandsage.com/google-gemini-statistics/ [Pages 1 and 20.]

[5] B. A. Becker, P. Denny, J. Finnie-Ansley, A. Luxton-Reilly, J. Prather, and E. A. Santos, "Programming is hard - or at least it used to be: Educational opportunities and challenges of ai code generation," in *Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1*, ser. SIGCSE 2023. New York, NY, USA: Association for Computing Machinery, 2023. doi: 10.1145/3545945.3569759. ISBN 9781450394314 p. 500–506. [Online]. Available: https://doi.org/10.1145/3545945.3569759 [Page 1.]

[6] Intelligent, "One-third of college students used chatgpt for schoolwork during the 2022-23 academic year," 2023. [Online]. Available: https://www.intelligent.com/one-third-of-college-students-used-chatgpt-for-schoolwork-during-the-2022-23-academic-year [Pages 1, 2, 10, and 11.]

[7] M. Sullivan, A. Kelly, and P. McLaughlan, "Chatgpt in higher education: Considerations for academic integrity and student learning," vol. 6, 03 2023. doi: 10.37074/jalt.2023.6.1.17 [Page 1.]

[8] P. Sharma and M. Harkishan, "Designing an intelligent tutoring system for computer programing in the pacific," *Education and Information Technologies*, vol. 27, no. 5, pp. 6197–6209, Jun 2022. doi: 10.1007/s10639-021-10882-9. [Online]. Available: https://doi.org/10.1007/s10639-021-10882-9 [Page 1.]

[9] L. Huang, H. Zhang, R. Li, Y. Ge, and J. Wang, "Ai coding: Learning to construct error correction codes," *IEEE Transactions on Communications*, vol. 68, no. 1, pp. 26–39, Jan 2020. doi: 10.1109/TCOMM.2019.2951403 [Page 1.]

[10] N. Kiesler and D. Schiffner, "Large language models in introductory programming education: Chatgpt's performance and implications for assessments," 2023. [Page 1.]

[11] N. Dunder, S. Lundborg, J. Wong, and O. Viberg, "Kattis vs chatgpt: Assessment and evaluation of programming tasks in the age of artificial intelligence," in *Proceedings of the 14th Learning Analytics and Knowledge Conference*, ser. LAK '24. New York, NY, USA: Association for Computing Machinery, 2024. doi: 10.1145/3636555.3636882. ISBN 9798400716188 p. 821–827. [Online]. Available: https://doi.org/10.1145/3636555.3636882 [Pages 1, 8, 9, 14, 15, 34, and 35.]

[12] C. Geng, Y. Zhang, B. Pientka, and X. Si, "Can chatgpt pass an introductory level functional language programming course?" 2023. [Pages 1, 8, 12, 14, 15, 34, and 35.]

[13] F. Liu, Y. Liu, L. Shi, H. Huang, R. Wang, Z. Yang, and L. Zhang, "Exploring and evaluating hallucinations in llm-powered code generation," 2024. [Page 2.]

[14] V. Rawte, S. Chakraborty, A. Pathak, A. Sarkar, S. M. T. I. Tonmoy, A. Chadha, A. P. Sheth, and A. Das, "The troubling emergence of hallucination in large language models – an extensive definition, quantification, and prescriptive remediations," 2023. [Pages 2, 10, and 11.]

[15] P. P. Ray, "Chatgpt: A comprehensive review on background, applications, key challenges, bias, ethics, limitations and future scope," *Internet of Things and Cyber-Physical Systems*, vol. 3, pp. 121–154, 2023. doi: https://doi.org/10.1016/j.iotcps.2023.04.003. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S266734522300024X [Pages 2 and 11.]

[16] S. M. T. I. Tonmoy, S. M. M. Zaman, V. Jain, A. Rani, V. Rawte, A. Chadha, and A. Das, "A comprehensive survey of hallucination mitigation techniques in large language models," 2024. [Pages 2, 11, 12, 15, 16, and 34.]

[17] Z. Jiang, F. Xu, L. Gao, Z. Sun, Q. Liu, J. Dwivedi-Yu, Y. Yang, J. Callan, and G. Neubig, "Active retrieval augmented generation," in *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, H. Bouamor, J. Pino, and K. Bali, Eds. Singapore: Association for Computational Linguistics, Dec. 2023. doi: 10.18653/v1/2023.emnlp-main.495 pp. 7969–7992. [Online]. Available: https://aclanthology.org/2023.emnlp-main.495 [Pages 2, 11, and 12.]

[18] P. Sahoo, A. K. Singh, S. Saha, V. Jain, S. Mondal, and A. Chadha, "A systematic survey of prompt engineering in large language models: Techniques and applications," 2024. [Pages 2, 11, 12, and 15.]

[19] Promptingguide.ai, "Retrieval augmented generation," 2021. [Online]. Available: https://www.promptingguide.ai/techniques/rag [Pages 2, 11, and 13.]

[20] O. Khattab, K. Santhanam, X. L. Li, D. Hall, P. Liang, C. Potts, and M. Zaharia, "Demonstrate-search-predict: Composing retrieval and language models for knowledge-intensive nlp," 2023. [Page 2.]

[21] Kattis, "Kattis," 2024. [Online]. Available: https://www.kattis.com/ [Pages 2, 8, 9, and 14.]

[22] N. Iqbal, H. Ahmed, and K. Azhar, "Exploring teachers' attitudes towards using chat gpt," *Global Journal for Management and Administrative Sciences*, vol. 3, 02 2023. doi: 10.46568/gjmas.v3i4.163 [Pages 5, 6, and 10.]

[23] C. K. Y. Chan and W. Hu, "Students' voices on generative ai: Perceptions, benefits, and challenges in higher education," 2023. [Pages 5, 6, 7, and 10.]

[24] P. Wulff, L. Mientus, A. Nowak, and A. Borowski, "Utilizing a pretrained language model (bert) to classify preservice physics teachers' written reflections," pp. 439–466, Sep 2023. [Online]. Available: https://doi.org/10.1007/s40593-022-00290-6 [Page 5.]

[25] M. Montenegro-Rueda, J. Fernández Cerero, J. Fernández Batanero, and E. Meneses, "Impact of the implementation of chatgpt in education: A systematic review," *Computers*, vol. 12, p. 153, 07 2023. doi: 10.3390/computers12080153 [Page 5.]

[26] C. Kooli, "Chatbots in education and research: A critical examination of ethical implications and solutions," *Sustainability*, vol. 15, no. 7, 2023. doi: 10.3390/su15075614. [Online]. Available: https://www.mdpi.com/2071-1050/15/7/5614 [Pages 6 and 8.]

[27] M. B. Malik Sallam, Nesreen A. Salim and A. B. Al-Tammemi, "Chatgpt applications in medical, dental, pharmacy, and public health education: A descriptive study highlighting the advantages and limitations," *Narra j*, vol. 3, no. 1, 2023. doi: 10.52225/narra.v3i1.103. [Online]. Available: https://narraj.org/main/article/view/103 [Pages 6 and 8.]

[28] S. H. Allehyani and M. A. Algamdi, "Digital competences: Early childhood teachers' beliefs and perceptions of chatgpt application in teaching english as a second language (esl)," *International journal of learning, teaching and educational research*, vol. 22, no. 11, pp. 343–363, 2023. [Pages 6 and 8.]

[29] A. Stojanov, "Learning with chatgpt 3.5 as a more knowledgeable other: An autoethnographic study," *International Journal of Educational Technology in Higher Education*, vol. 20, no. 1, p. 35, 2023. [Pages 6 and 8.]

[30] P. Limna, T. Kraiwanit, K. Jangjarat, P. Klayklung, and P. Chocksathaporn, "The use of chatgpt in the digital era: Perspectives on chatbot implementation," *Journal of Applied Learning and Teaching*, vol. 6, no. 1, 2023. [Page 6.]

[31] G. Kiryakova and N. Angelova, "Chatgpt—a challenging tool for the university professors in their teaching practice," *Education Sciences*, vol. 13, no. 10, p. 1056, 2023. [Page 6.]

[32] M. C. Keiper, G. Fried, J. Lupinek, and H. Nordstrom, "Artificial intelligence in sport management education: Playing the ai game with chatgpt," *Journal of Hospitality, Leisure, Sport & Tourism Education*, vol. 33, p. 100456, 2023. [Page 6.]

[33] G. van den Berg and E. du Plessis, "Chatgpt and generative ai: Possibilities for its contribution to lesson planning, critical thinking and openness in teacher education," *Education Sciences*, vol. 13, no. 10, p. 998, 2023. [Page 6.]

[34] A. Lozano and C. Blanco Fontao, "Is the education system prepared for the irruption of artificial intelligence? a study on the perceptions of students of primary education degree from a dual perspective: Current pupils and future teachers," *Education Sciences*, vol. 13, no. 7, p. 733, 2023. [Page 6.]

[35] A. Strawhacker and M. U. Bers, "What they learn when they learn coding: investigating cognitive domains and computer programming knowledge in young children," *Educational Technology Research and Development*, vol. 67, no. 3, pp. 541–575, Jun 2019. doi: 10.1007/s11423-018-9622-x. [Online]. Available: https://doi.org/10.1007/s11423-018-9622-x [Page 7.]

[36] R. MATHEW, S. I. MALIK, and R. M. TAWAFAK, "Teaching problem solving skills using an educational game in a computer programming course," *Informatics in Education*, vol. 18, no. 2, pp. 359–373, 2019. doi: 10.15388/infedu.2019.17 [Page 7.]

[37] Y.-S. Su, M. Shao, and L. Zhao, "Effect of mind mapping on creative thinking of children in scratch visual programming education," *Journal of Educational Computing Research*, vol. 60, no. 4, pp. 906–929, 2022. doi: 10.1177/07356331211053383. [Online]. Available: https://doi.org/10.1177/07356331211053383 [Page 7.]

[38] I. Milne and G. Rowe, "Difficulties in learning and teaching programming—views of students and tutors," *Education and Information Technologies*, vol. 7, no. 1, pp. 55–66, Mar 2002. doi: 10.1023/A:1015362608943. [Online]. Available: https://doi.org/10.1023/A:1015362608943 [Page 7.]

[39] T. Phung, V.-A. Pădurean, J. Cambronero, S. Gulwani, T. Kohn, R. Majumdar, A. Singla, and G. Soares, "Generative ai for programming education: Benchmarking chatgpt, gpt-4, and human tutors," 2023. [Page 8.]

[40] D. T. T. Mai, C. V. Da, and N. V. Hanh, "The use of chatgpt in teaching and learning: a systematic review through swot analysis approach," *Frontiers in Education*, vol. 9, 2024. doi: 10.3389/feduc.2024.1328769. [Online]. Available: https://www.frontiersin.org/articles/10.3389/feduc.2024.1328769 [Page 8.]

[41] J. Savelka, A. Agarwal, M. An, C. Bogart, and M. Sakr, "Thrilled by your progress! large language models (gpt-4) no longer struggle to pass assessments in higher education programming courses," in *Proceedings of the 2023 ACM Conference on International Computing Education Research V.1*, ser. ICER 2023. ACM, Aug. 2023. doi: 10.1145/3568813.3600142. [Online]. Available: http://dx.doi.org/10.1145/3568813.3600142 [Page 8.]

[42] B. Fernandez-Gauna, N. Rojo, and M. Graña, "Automatic feedback and assessment of team-coding assignments in a devops context," *International Journal of Educational Technology in Higher Education*, vol. 20, 03 2023. doi: 10.1186/s41239-023-00386-6 [Page 8.]

[43] S. Wasik, M. Antczak, J. Badura, A. Laskowski, and T. Sternal, "A survey on online judge systems and their applications," *ACM Comput. Surv.*, vol. 51, no. 1, jan 2018. doi: 10.1145/3143560. [Online]. Available: https://doi.org/10.1145/3143560 [Pages 8 and 9.]

[44] J. Petit, O. Giménez, and S. Roura, "Jutge.org: an educational programming judge," 02 2012. doi: 10.1145/2157136.2157267 [Page 9.]

[45] R. Miguel A, M. Shahriar, and L. Rujia, "Competitive learning in informatics: The uva online judge experience," *Olympiads in Informatics 2*, 2008. [Page 9.]

[46] E. Enström, G. Kreitz, F. Niemelä, P. Söderman, and V. Kann, "Five years with kattis — using an automated assessment system in teaching," in *2011 Frontiers in Education Conference (FIE)*, 2011. doi: 10.1109/FIE.2011.6142931 pp. T3J–1–T3J–6. [Page 9.]

[47] Kattis, "Kattis problem archive," 2024. [Online]. Available: https://open.kattis.com/ [Pages xi, 9, 10, 21, 22, and 23.]

[48] D. O. Eke, "Chatgpt and the rise of generative ai: Threat to academic integrity?" *Journal of Responsible Technology*, vol. 13, p. 100060, 2023. doi: https://doi.org/10.1016/j.jrt.2023.100060. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2666659623000033 [Page 10.]

[49] L. Uzun, "Chatgpt and academic integrity concerns: Detecting artificial intelligence generated content," vol. 3, pp. 45–54, 04 2023. [Page 10.]

[50] J. Wang, S. Liu, X. Xie, and Y. Li, "Evaluating aigc detectors on code content," 2023. [Page 10.]

[51] W. H. Pan, M. J. Chok, J. L. S. Wong, Y. X. Shin, Y. S. Poon, Z. Yang, C. Y. Chong, D. Lo, and M. K. Lim, "Assessing ai detectors in identifying ai-generated code: Implications for education," 2024. [Page 10.]

[52] Promptingguide.ai, "Few-shot prompting," 2021. [Online]. Available: https://www.promptingguide.ai/techniques/fewshot [Pages 11 and 12.]

[53] Y. Tao, O. Viberg, R. S. Baker, and R. F. Kizilcec, "Auditing and mitigating cultural bias in llms," 2023. [Page 12.]

[54] B. Chen, Z. Zhang, N. Langrené, and S. Zhu, "Unleashing the potential of prompt engineering in large language models: a comprehensive review," 2023. [Page 12.]

[55] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei, "Language models are few-shot learners," 2020. [Page 12.]

[56] S. Riedel, D. Kiela, P. Lewis, and A. Piktus, "Retrieval augmented generation: Streamlining the creation of intelligent natural language processing models," 2020. [Online]. Available: https://ai.meta.com/blog/retrieval-augmented-generation-streamlining-the-creation-of-intelligent-natural-language-processing-models/ [Page 13.]

[57] T. Kwiatkowski, J. Palomaki, O. Redfield, M. Collins, A. Parikh, C. Alberti, D. Epstein, I. Polosukhin, M. Kelcey, J. Devlin, K. Lee, K. N. Toutanova, L. Jones, M.-W. Chang, A. Dai, J. Uszkoreit, Q. Le, and S. Petrov, "Natural questions: a benchmark for question answering research," *Transactions of the Association of Computational Linguistics*, 2019. [Page 13.]

[58] A. Bordes, S. Chopra, and J. Weston, "Question answering with subgraph embeddings," 2014. [Page 13.]

[59] P. Baudiš and J. Šedivý, "Modeling of the question answering task in the yodaqa system," 09 2015. doi: 10.1007/978-3-319-24027-$5_20.ISBN978-3-319-24026-8pp.222--228.[Page$ 13.]

[60] L. Murr, M. Grainger, and D. Gao, "Testing llms on code generation with varying levels of prompt specificity," 2023. [Pages 14 and 15.]

[61] F. Hans, "Chatgpt vs. bard - which is better at solving coding problems?" 08 2023. [Pages 14, 15, and 33.]

[62] The AlphaCode team, "Competitive programming with alphacode," Dec 2022. [Online]. Available: https://deepmind.google/discover/blog/competitive-programming-with-alphacode/ [Pages 15 and 34.]

[63] Github, "The world's most widely adopted ai developer tool." Oct 2021. [Online]. Available: https://github.com/features/copilot [Pages 15, 34, and 35.]

[64] W. Zaremba and G. Brockman, "Openai codex," Aug 2021. [Online]. Available: https://openai.com/index/openai-codex [Pages 15 and 34.]

[65] J. Finnie-Ansley, P. Denny, B. A. Becker, A. Luxton-Reilly, and J. Prather, "The robots are coming: Exploring the implications of openai codex on introductory programming," in *Proceedings of the 24th Australasian Computing Education Conference*, ser. ACE '22. New York, NY, USA: Association for Computing Machinery, 2022.

doi: 10.1145/3511861.3511863. ISBN 9781450396431 p. 10–19. [Online]. Available: https://doi.org/10.1145/3511861.3511863 [Pages 15 and 34.]

[66] Y. Li, D. Choi, J. Chung, N. Kushman, J. Schrittwieser, and R. Leblond, et al., "Competition-level code generation with alphacode," *Science*, vol. 378, no. 6624, pp. 1092–1097, 2022. doi: 10.1126/science.abq1158. [Online]. Available: https://www.science.org/doi/abs/10.1126/science.abq1158 [Pages 15 and 34.]

[67] N. Nguyen and S. Nadi, "An empirical evaluation of github copilot's code suggestions," in *Proceedings of the 19th International Conference on Mining Software Repositories*, ser. MSR '22. New York, NY, USA: Association for Computing Machinery, 2022. doi: 10.1145/3524842.3528470. ISBN 9781450393034 p. 1–5. [Online]. Available: https://doi.org/10.1145/3524842.3528470 [Pages 15 and 34.]

[68] 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, A. Yazdanbakhsh, and P. Clark, "Self-refine: Iterative refinement with self-feedback," 2023. [Page 15.]

[69] K. Martineau, "What is prompt-tuning?" 2023. [Online]. Available: https://research.ibm.com/blog/what-is-ai-prompt-tuning [Page 15.]

[70] H. Su, S. Jiang, Y. Lai, H. Wu, B. Shi, C. Liu, Q. Liu, and T. Yu, "Arks: Active retrieval in knowledge soup for code generation," 2024. [Pages ix, 16, and 34.]

[71] OpenAI, "Chatgpt: Optimizing language models for dialogue," 2023. [Online]. Available: https://openai.com/blog/chatgpt [Pages 20, 22, and 23.]

[72] R. Anil, S. Borgeaud, W. Yonghui, J. B. Alayrac, J. Yu, and R. Soricut, et al., "Gemini: A family of highly capable multimodal models," 2023. [Pages xi, 20, 23, and 35.]

[73] D. Hendrycks, C. Burns, S. Basart, A. Zou, M. Mazeika, D. Song, and J. Steinhardt, "Measuring massive multitask language understanding," 2021. [Page 20.]

[74] R. Zellers, A. Holtzman, Y. Bisk, A. Farhadi, and Y. Choi, "Hellaswag: Can a machine really finish your sentence?" 2019. [Page 20.]

[75] D. Hendrycks, C. Burns, S. Kadavath, A. Arora, S. Basart, E. Tang, D. Song, and J. Steinhardt, "Measuring mathematical problem solving with the math dataset," 2021. [Page 20.]

[76] Y. Hao, G. Li, Y. Liu, X. Miao, H. Zong, S. Jiang, Y. Liu, and H. Wei, "Aixbench: A code generation benchmark dataset," 2022. [Page 20.]

[77] M. Chen, J. Tworek, H. Jun, Q. Yuan, H. P. de Oliveira Pinto, and J. Kaplan, et al., "Evaluating large language models trained on code," 2021. [Page 20.]

[78] D. Dua, Y. Wang, P. Dasigi, G. Stanovsky, S. Singh, and M. Gardner, "Drop: A reading comprehension benchmark requiring discrete reasoning over paragraphs," 2019. [Page 20.]

[79] OpenAI, "Gpt-4," 2023. [Online]. Available: https://openai.com/research/gpt-4 [Pages xi and 20.]

[80] K. G. A. Ludvigsen, "The carbon footprint of chatgpt," Dec 2022. [Online]. Available: https://towardsdatascience.com/the-carbon-footprint-of-chatgpt-66932314627d [Page 36.]

[81] P. Li, J. Yang, M. A. Islam, and S. Ren, "Making ai less "thirsty": Uncovering and addressing the secret water footprint of ai models," 2023. [Page 36.]

Appendix A

Breakdown of Experiment

This section breaks down the experiment method that was described in Chapter 3. Firstly, a table listing the names of the 100 assignments used during the experiment along with their difficulty and some relevant metadata from Kattis. The assignments can all be found on Kattis by searching for their names. Secondly, a link to the prompts created for the experiment is given in Section A.2.

A.1 Assignments Choosen for Experiment

Table A.1: All assignments chosen for the experiment

Name	Difficulty	Submissions	Accepted submission
Quadrant Selection	1.2	64706	35146
Add Two Numbers	1.2	29613	16273
Autori	1.2	52688	29421
R2	1.3	66376	36998
Bitte ein Bit	1.3	1750	1044
Two-sum	1.3	24242	12676
Blandað Best	1.3	485	289
Betting	1.3	7745	4803

Continued on next page

Table A.1: All assignments chosen for the experiment (Continued)

Name	Difficulty	Submissions	Accepted submission
Chanukah Challenge	1.4	11656	6851
Solving for Carrots	1.4	56524	34456
Reversed Binary Numbers	1.5	23004	13833
Karte	1.6	5306	3033
Aaah!	1.7	52754	23110
Sjecista	2	2404	1536
Moscow Dream	2	12414	3924
Exactly Electrical	2.1	3685	1558
Hot Hike	2.1	5582	2824
Above Average	2.3	23150	8593
Army Strength (Easy)	2.3	5546	2717
Soft Passwords	2.4	3393	1188
Math Homework	2.4	7448	3097
Akcija	2.5	17657	7508
Bazen	2.5	1397	544
The Weight Of Words	2.5	2583	972
Closest Sums	2.6	9095	3221
ABC	2.7	34173	14065
3D Printed Statues	2.7	25556	10159
Supercomputer	2.8	8983	3211
Baloni	3	9201	3033
Multigram	3	1133	542
Birds on a Wire	3.3	6949	2017

Continued on next page

Table A.1: All assignments chosen for the experiment (Continued)

Name	Difficulty	Submissions	Accepted submission
Kolone	3.4	1288	580
Book Shelves	3.5	2555	580
Eating Out	3.5	4607	1242
House of Cards	3.5	1941	724
Arctic Network	3.9	3693	1273
Where to Live?	4	846	313
Diagonal Cut	4	3323	1065
Adding Words	4.1	21522	5232
10 Kinds of People	4.1	38162	7292
A1 Paper	4.2	9190	2698
Alphabet Animals	4.2	12529	2234
Euclid's Game	4.3	1953	620
Freckles	4.5	5095	1243
Cakey McCakeFace	4.7	2810	894
Game Night	4.9	451	192
Conservation	4.9	2636	805
Wolf	4.9	501	166
Thesaurus	5	1027	298
Trending Topic	5.1	1482	457
Unique Dice	5.1	1028	330
Sensor Network	5.2	149	43
Antenna Analysis	5.4	1547	471
Unown Code (Easy)	5.4	116	39
Stol	5.6	458	166
Wedding	5.6	527	101
Abridged Reading	5.7	1071	311

Continued on next page

Table A.1: All assignments chosen for the experiment (Continued)

Name	Difficulty	Submissions	Accepted submission
Lemonade Trade	5.8	1169	223
Wipe Your Whiteboards	5.8	447	134
0-1 Sequences	5.9	19107	2971
Bag of Tiles	6	1827	570
Taxi Cab Scheme	6.1	313	125
Global Warming	6.2	481	165
Biggest Slice	6.2	1752	410
EvenOdd	6.3	839	214
Max Arithmetic Subsequence	7.1	269	48
Number Sets (Hard)	7.1	1026	311
Radar	7.1	239	47
String Multimatching	7.1	5375	1312
Colliding Traffic	7.2	855	137
Sumsets	7.2	5075	870
Equations	7.3	701	166
Dvoniz	7.3	391	115
Liga	7.3	1832	177
Money Transfers	7.3	288	61
Product Divisors	7.3	352	87
Partial Linear Equation Solver	7.5	1997	385
Binary search tree	7.6	7537	1611
Gas Station Numbers	7.7	141	27

Continued on next page

Table A.1: All assignments chosen for the experiment (Continued)

Name	Difficulty	Submissions	Accepted submission
Lone Rook	7.7	522	95
Cycles (Hard)	8	316	93
Icons in the Toolbar	8	190	86
Division	8.1	729	109
Endless Knight	8.2	344	68
GCD Sum	8.2	276	44
Guess the Numbers	8.2	759	177
Basic	8.4	594	105
3-Sided Dice	8.5	4070	372
Crusaders of the Lost Mark	8.5	1279	145
Kolkrabbaleikarnir	8.5	193	162
1-D Frogger (Hard)	8.6	4351	354
Guess the Digits	8.6	778	44
Mravi	8.8	552	99
Closeness Queries	8.9	282	194
Frumtölutalning	9.1	377	276
A Different List Game	9.1	8445	595
Magical Mystery Knight's Tour	9.3	1775	86
Towers of Powers 2: Power Harder	9.4	7927	420
Travelling Salesperson 2D	9.5*	115101	75461
Uuu	9.8	392	219

A.2 Initial Prompts Created for Experiment

The prompts created utilizing RAG are available online in an OSF project. The reason for only the RAG prompts being available is because the non-RAG prompts only consisted of the assignment description available on Kattis for each assignment, as specified in Section 3.5.1. Following the link below will take you to the project, which consists of a folder containing all the prompts in the form of text files. Each file containing a prompt is named after the assignment the prompt was meant for, totalling 100 text files.

Link to the prompts: https://osf.io/t82cn/?view_only=df68c812df48432d8e7fb16dc6022f8d

Appendix B

Breakdown of Results

This section breaks down the experiment results presented in Chapter 4 into four tables. Each table presents the exact response returned by Kattis to each attempt at solving the 100 assignments listed in Appendix A. Each table represents the individual result of either ChatGPT or Gemini when utilizing RAG and without it. The six possible messages returned by Kattis are listed in Figure B.1 along with their abbreviated form that will be used henceforth in this chapter.

Accepted (AC)	The submission successfully passed all test cases.
Wrong Answer (WA)	The submission output did not match the expected output of atleast one test case.
Run Time Error (RTE)	The submission threw an error during its run time during atleast one test case.
Compile Error (CE)	The submission did not compile correctly.
Time Limit Exceeded (TLE)	The submission took too long to solve atleast one test case.
Memory Limit Exceeded (MLE)	The submission used too much memory to solve atleast one test case.

Figure B.1: Distribution of Kattis responses for each submission separated based on LLM and usage of RAG.

In the following four tables, a response code can be followed by a fraction within parenthesis, e.g., ”WA (15/74)” or ”RTE (8/25)”. This means that the solution submitted by the model passed some test cases on Kattis. The

numerator represents the number of test cases passed, and the denominator represents the total number of test cases. In such cases, the response code presented in the table was returned for the test case after the one that was passed. In case no test cases were passed no fraction will be written in the cell, instead only the response code returned by Kattis on the first test case will be written.

The tables consist of six columns, the name of the assignment attempted followed by five columns for each of the five possible attempts a model can have to solve the assignment. Not every assignment was attempted five times. How an assignment's number of attempts was calculated can be read about in Section 3.4. In case an attempt is accepted, all the following cells in the row will contain the value "N/A", which means not attempted. This is also the case when the maximum amount of attempts has been reached for an assignment. Lastly, this chapter presents a link to all the code generated throughout the experiment in Section B.5.

B.1 ChatGPT Without RAG Results

Table B.1: Detailed results of ChatGPT without RAG

Name	1st	2nd	3rd	4th	5th
Quadrant Selection	AC	N/A	N/A	N/A	N/A
Add Two Numbers	AC	N/A	N/A	N/A	N/A
Autori	AC	N/A	N/A	N/A	N/A
R2	AC	N/A	N/A	N/A	N/A
Bitte ein Bit	AC	N/A	N/A	N/A	N/A
Two-sum	WA	N/A	N/A	N/A	N/A
Blandað Best	AC	N/A	N/A	N/A	N/A
Betting	AC	N/A	N/A	N/A	N/A

Continued on next page

Table B.1: Detailed results of ChatGPT without RAG (Continued)

Name	1st	2nd	3rd	4th	5th
Chanukah Challenge	AC	N/A	N/A	N/A	N/A
Solving for Carrots	WA	N/A	N/A	N/A	N/A
Reversed Binary Numbers	AC	N/A	N/A	N/A	N/A
Travelling Salesperson 2D	AC	N/A	N/A	N/A	N/A
Karte	AC	N/A	N/A	N/A	N/A
Aaah!	AC	N/A	N/A	N/A	N/A
Sjecista	WA (1/13)	N/A	N/A	N/A	N/A
Moscow Dream	WA (5/28)	WA (5/28)	WA (7/28)	N/A	N/A
Exactly Electrical	AC	N/A	N/A	N/A	N/A
Hot Hike	WA	WA	N/A	N/A	N/A
Above Average	WA	AC	N/A	N/A	N/A
Army Strength (Easy)	RTE	RTE	N/A	N/A	N/A
Soft Passwords	AC	N/A	N/A	N/A	N/A
Math Homework	WA	WA	N/A	N/A	N/A
Akcija	AC	N/A	N/A	N/A	N/A
Bazen	WA	WA	N/A	N/A	N/A
The Weight Of Words	AC	N/A	N/A	N/A	N/A
Closest Sums	AC	N/A	N/A	N/A	N/A
ABC	AC	N/A	N/A	N/A	N/A

Continued on next page

Table B.1: Detailed results of ChatGPT without RAG (Continued)

Name	1st	2nd	3rd	4th	5th
3D Printed Statues	WA	WA	N/A	N/A	N/A
Supercomputer	TLE (3/13)	WA	TLE (3/13)	N/A	N/A
Baloni	WA (2/13)	WA	WA	N/A	N/A
Multigram	WA	WA	N/A	N/A	N/A
Birds on a Wire	WA	WA	WA	N/A	N/A
Kolone	WA	WA	N/A	N/A	N/A
Book Shelves	WA (2/52)	WA (2/52)	CE	WA (2/52)	WA (2/52)
Eating Out	WA	WA	WA	N/A	N/A
House of Cards	WA	WA	N/A	N/A	N/A
Arctic Network	WA (1/2)	WA (1/2)	RTE	N/A	N/A
Where to Live?	WA (4/5)	WA (4/5)	N/A	N/A	N/A
Diagonal Cut	WA	WA	N/A	N/A	N/A
Adding Words	WA	WA	WA	WA	N/A
10 Kinds of People	WA	WA	WA	WA	WA (1/25)
A1 Paper	WA	WA	WA	N/A	N/A
Alphabet Animals	WA	WA	WA	WA	WA
Euclid's Game	RTE (1/2)	RTE (1/2)	TLE (1/2)	N/A	N/A
Freckles	TLE (1/2)	RTE (1/2)	RTE (1/2)	RTE (1/2)	N/A
Cakey McCakeFace	WA (1/15)	WA (1/15)	WA	N/A	N/A

Continued on next page

Table B.1: Detailed results of ChatGPT without RAG (Continued)

Name	1st	2nd	3rd	4th	5th
Game Night	WA	WA	N/A	N/A	N/A
Conservation	WA	RTE	WA	WA	N/A
Wolf	WA	WA	WA (2/31)	N/A	N/A
Thesaurus	WA	WA	WA	N/A	N/A
Trending Topic	WA (1/5)	WA (1/5)	WA (1/5)	WA	N/A
Unique Dice	WA (1/44)	WA (1/44)	WA (1/44)	N/A	N/A
Sensor Network	RTE	RTE	RTE	N/A	N/A
Antenna Analysis	WA	WA	N/A	N/A	N/A
Unown Code (Easy)	TLE (2/19)	WA	N/A	N/A	N/A
Stol	WA	WA	WA	N/A	N/A
Wedding	WA	RTE	WA	WA	WA
Abridged Reading	RTE	RTE	WA	N/A	N/A
Lemonade Trade	WA (1/15)	WA (1/15)	WA (1/15)	WA	N/A
Wipe Your Whiteboards	TLE	TLE	TLE	N/A	N/A
0-1 Sequences	WA	WA	WA	WA	N/A
Bag of Tiles	TLE (1/4)	TLE (1/4)	TLE (1/4)	TLE (1/4)	N/A
Taxi Cab Scheme	RTE	WA	WA	N/A	N/A
Global Warming	WA (1/57)	WA	WA	N/A	N/A
Biggest Slice	WA	WA	WA	WA	N/A

Continued on next page

Table B.1: Detailed results of ChatGPT without RAG (Continued)

Name	1st	2nd	3rd	4th	5th
EvenOdd	MLE (2/17)	TLE	N/A	N/A	N/A
Max Arithmetic Subsequence	TLE (3/14)	WA	TLE (3/14)	TLE (3/14)	WA
Number Sets (Hard)	WA	WA	WA	N/A	N/A
Radar	WA (1/17)	WA	WA	TLE (1/17)	WA
String Multimatching	WA	WA	WA	WA	WA
Colliding Traffic	WA	WA	WA	RTE	WA
Sumsets	TLE (5/33)	TLE (5/33)	WA (5/33)	WA (2/33)	WA (2/33)
Equations	RTE	RTE	RTE	RTE	N/A
Dvoniz	RTE	RTE	RTE	N/A	N/A
Liga	WA	WA	RTE	RTE	WA
Money Transfers	WA	WA	WA	WA	WA
Product Divisors	WA	WA	TLE (4/16)	WA	N/A
Partial Linear Equation Solver	WA	WA	RTE	WA	WA
Binary search tree	WA	WA	WA	RTE	N/A
Gas Station Numbers	WA	WA	WA	WA	N/A
Lone Rook	RTE	RTE	RTE	RTE	WA
Cycles (Hard)	WA	WA	WA	N/A	N/A
Icons in the Toolbar	WA	WA	N/A	N/A	N/A
Division	RTE	RTE	RTE	RTE	RTE

Continued on next page

Table B.1: Detailed results of ChatGPT without RAG (Continued)

Name	1st	2nd	3rd	4th	5th
Endless Knight	MLE (1/3)	TLE (1/3)	RTE	TLE (1/3)	N/A
GCD Sum	TLE (4/48)	TLE (4/48)	TLE (4/48)	TLE (4/48)	N/A
Guess the Numbers	RTE	RTE	RTE	N/A	N/A
Basic	WA	WA	WA	WA	N/A
3-Sided Dice	RTE	RTE	RTE	RTE	RTE
Crusaders of the Lost Mark	WA	WA	WA	WA	WA
Kolkrabbaleikarnir	WA (9/154)	WA (21/154)	WA (21/154)	WA (4/154)	N/A
1-D Frogger (Hard)	WA	WA	WA	RTE	WA
Guess the Digits	WA	WA	WA	WA	WA
Mravi	RTE	RTE	RTE	RTE	RTE
Closeness Queries	TLE (11/21)	CE	CE	CE	N/A
Frumtölutalning	WA (56/124)	WA (78/124)	WA (78/124)	WA (56/124)	WA (78/124)
A Different List Game	WA	WA	WA	WA	WA
Magical Mystery Knight's Tour	TLE	TLE	TLE	TLE	TLE
Towers of Powers 2: Power Harder	RTE	RTE	RTE	RTE	RTE
Uuu	WA	WA	WA	N/A	N/A

B.2 ChatGPT With RAG Results

Table B.2: Detailed results of ChatGPT with RAG

Name	1st	2nd	3rd	4th	5th
Quadrant Selection	AC	N/A	N/A	N/A	N/A
Add Two Numbers	AC	N/A	N/A	N/A	N/A
Autori	AC	N/A	N/A	N/A	N/A
R2	AC	N/A	N/A	N/A	N/A
Bitte ein Bit	AC	N/A	N/A	N/A	N/A
Two-sum	AC	N/A	N/A	N/A	N/A
Blandað Best	AC	N/A	N/A	N/A	N/A
Betting	AC	N/A	N/A	N/A	N/A
Chanukah Challenge	AC	N/A	N/A	N/A	N/A
Solving for Carrots	AC	N/A	N/A	N/A	N/A
Reversed Binary Numbers	AC	N/A	N/A	N/A	N/A
Travelling Salesperson 2D	AC	N/A	N/A	N/A	N/A
Karte	AC	N/A	N/A	N/A	N/A
Aaah!	AC	N/A	N/A	N/A	N/A
Sjecista	AC	N/A	N/A	N/A	N/A
Moscow Dream	AC	N/A	N/A	N/A	N/A
Exactly Electrical	AC	N/A	N/A	N/A	N/A
Hot Hike	WA	WA	N/A	N/A	N/A
Above Average	AC	N/A	N/A	N/A	N/A
Army Strength (Easy)	AC	N/A	N/A	N/A	N/A
Soft Passwords	AC	N/A	N/A	N/A	N/A

Continued on next page

Table B.2: Detailed results of ChatGPT with RAG (Continued)

Name	1st	2nd	3rd	4th	5th
Math Homework	WA	WA	N/A	N/A	N/A
Akcija	AC	N/A	N/A	N/A	N/A
Bazen	WA (1/13)	WA (1/13)	N/A	N/A	N/A
The Weight Of Words	AC	N/A	N/A	N/A	N/A
Closest Sums	AC	N/A	N/A	N/A	N/A
ABC	AC	N/A	N/A	N/A	N/A
3D Printed Statues	WA (1/33)	WA (1/33)	N/A	N/A	N/A
Supercomputer	TLE (3/13)	WA	WA	N/A	N/A
Baloni	WA (2/13)	WA (2/13)	CE	N/A	N/A
Multigram	WA	WA	N/A	N/A	N/A
Birds on a Wire	RTE (1/11)	AC	N/A	N/A	N/A
Kolone	WA	WA	N/A	N/A	N/A
Book Shelves	WA (2/52)	WA (2/52)	WA (2/52)	WA (3/52)	WA
Eating Out	AC	N/A	N/A	N/A	N/A
House of Cards	WA	WA	N/A	N/A	N/A
Arctic Network	RTE	RTE	RTE	N/A	N/A
Where to Live?	TLE (2/5)	WA (4/5)	N/A	N/A	N/A
Diagonal Cut	AC	N/A	N/A	N/A	N/A
Adding Words	WA	WA	WA	WA	N/A
10 Kinds of People	RTE	WA (1/25)	WA (1/25)	WA (1/25)	WA (1/25)

Continued on next page

Table B.2: Detailed results of ChatGPT with RAG (Continued)

Name	1st	2nd	3rd	4th	5th
A1 Paper	WA	WA	WA	N/A	N/A
Alphabet Animals	WA (2/35)	WA	AC	N/A	N/A
Euclid's Game	AC	N/A	N/A	N/A	N/A
Freckles	TLE (1/2)	TLE (1/2)	TLE (1/2)	TLE (1/2)	N/A
Cakey McCakeFace	AC	N/A	N/A	N/A	N/A
Game Night	WA	WA (2/33)	N/A	N/A	N/A
Conservation	WA	WA	WA	RTE	N/A
Wolf	WA (2/31)	WA (2/31)	WA (11/31)	N/A	N/A
Thesaurus	WA	WA	WA	N/A	N/A
Trending Topic	WA	WA	WA	WA	N/A
Unique Dice	WA (1/44)	WA (1/44)	WA	N/A	N/A
Sensor Network	RTE	RTE	RTE	N/A	N/A
Antenna Analysis	WA	WA	N/A	N/A	N/A
Unown Code (Easy)	WA	WA	N/A	N/A	N/A
Stol	RTE	WA (3/18)	WA (3/18)	N/A	N/A
Wedding	WA	WA	WA	RTE	WA
Abridged Reading	WA	WA	RTE	N/A	N/A
Lemonade Trade	WA	WA	WA	WA	N/A
Wipe Your Whiteboards	WA	WA	WA	N/A	N/A

Continued on next page

Table B.2: Detailed results of ChatGPT with RAG (Continued)

Name	1st	2nd	3rd	4th	5th
0-1 Sequences	AC	N/A	N/A	N/A	N/A
Bag of Tiles	TLE (1/4)	WA	WA	TLE (1/4)	N/A
Taxi Cab Scheme	WA	WA	WA	N/A	N/A
Global Warming	WA (1/57)	WA (1/57)	WA (1/57)	N/A	N/A
Biggest Slice	WA	WA	WA	WA	N/A
EvenOdd	TLE (2/17)	TLE (2/17)	N/A	N/A	N/A
Max Arithmetic Subsequence	WA	WA	WA	TLE (3/14)	WA
Number Sets (Hard)	WA	WA	WA	N/A	N/A
Radar	WA	WA	WA	WA	WA
String Multimatching	WA	WA	WA	WA	WA
Colliding Traffic	RTE	WA (2/6)	WA (2/6)	WA (2/6)	WA (2/6)
Sumsets	WA (2/33)	WA (2/33)	WA (2/33)	WA (2/33)	WA (2/33)
Equations	RTE	RTE	RTE	RTE	N/A
Dvoniz	WA	WA	WA	N/A	N/A
Liga	RTE	RTE	RTE	WA	WA
Money Transfers	RTE	WA	WA	WA	WA
Product Divisors	WA	WA	WA	WA	N/A
Partial Linear Equation Solver	WA	WA	WA	WA	WA
Binary search tree	WA	WA	TLE (4/13)	WA (1/13)	N/A

Continued on next page

Table B.2: Detailed results of ChatGPT with RAG (Continued)

Name	1st	2nd	3rd	4th	5th
Gas Station Numbers	RTE	RTE	RTE	RTE	N/A
Lone Rook	WA (1/47)	WA (1/47)	WA (3/47)	WA (3/47)	WA (1/47)
Cycles (Hard)	WA	WA	WA	N/A	N/A
Icons in the Toolbar	WA	WA	N/A	N/A	N/A
Division	WA	WA	WA	WA	WA
Endless Knight	MLE (1/3)	WA	WA	WA	N/A
GCD Sum	WA	WA	WA	WA	N/A
Guess the Numbers	RTE	RTE	RTE	N/A	N/A
Basic	WA	WA	WA	WA	N/A
3-Sided Dice	RTE	WA	WA	WA (1/2)	RTE
Crusaders of the Lost Mark	RTE	WA	WA	RTE	WA
Kolkrabbaleikarnir	RTE (1/154)	RTE (1/154)	WA (4/154)	WA (4/154)	N/A
1-D Frogger (Hard)	WA	WA	WA	WA	WA
Guess the Digits	RTE	WA	WA	WA	WA
Mravi	WA	WA	WA	RTE	RTE
Closeness Queries	TLE (11/21)	MLE (1/21)	TLE (11/21)	WA	N/A
Frumtölutalning	WA (56/124)	WA (104/124)	WA	CE	WA (104/124)
A Different List Game	WA	WA	WA	WA	WA

Continued on next page

Table B.2: Detailed results of ChatGPT with RAG (Continued)

Name	1st	2nd	3rd	4th	5th
Magical Mystery Knight's Tour	TLE	TLE	WA	WA	TLE
Towers of Powers 2: Power Harder	WA	WA	WA	WA	N/A
Uuu	WA	WA	WA	N/A	N/A

B.3 Gemini Without RAG Results

Table B.3: Detailed results of Gemini without RAG

Name	1st	2nd	3rd	4th	5th
Quadrant Selection	WA	N/A	N/A	N/A	N/A
Add Two Numbers	WA	RTE	N/A	N/A	N/A
Autori	AC	N/A	N/A	N/A	N/A
R2	AC	N/A	N/A	N/A	N/A
Bitte ein Bit	WA	N/A	N/A	N/A	N/A
Two-sum	RTE	N/A	N/A	N/A	N/A
Blandað Best	AC	N/A	N/A	N/A	N/A
Betting	WA	N/A	N/A	N/A	N/A
Chanukah Challenge	WA	N/A	N/A	N/A	N/A
Solving for Carrots	AC	N/A	N/A	N/A	N/A
Reversed Binary Numbers	AC	N/A	N/A	N/A	N/A
Travelling Salesperson 2D	AC	N/A	N/A	N/A	N/A

Continued on next page

Table B.3: Detailed results of Gemini without RAG (Continued)

Name	1st	2nd	3rd	4th	5th
Karte	RTE	N/A	N/A	N/A	N/A
Aaah!	AC	N/A	N/A	N/A	N/A
Sjecista	WA	N/A	N/A	N/A	N/A
Moscow Dream	WA (5/28)	WA (5/28)	WA (5/28)	N/A	N/A
Exactly Electrical	AC	N/A	N/A	N/A	N/A
Hot Hike	WA	WA	N/A	N/A	N/A
Above Average	RTE	RTE	N/A	N/A	N/A
Army Strength (Easy)	RTE	RTE	N/A	N/A	N/A
Soft Passwords	WA (3/38)	WA (3/38)	N/A	N/A	N/A
Math Homework	WA	RTE	N/A	N/A	N/A
Akcija	WA	WA	N/A	N/A	N/A
Bazen	WA	WA	N/A	N/A	N/A
The Weight Of Words	TLE (2/23)	TLE	TLE	N/A	N/A
Closest Sums	RTE	RTE	RTE	N/A	N/A
ABC	WA	AC	N/A	N/A	N/A
3D Printed Statues	WA	WA	N/A	N/A	N/A
Supercomputer	TLE (3/13)	TLE (3/13)	WA	N/A	N/A
Baloni	WA	WA	WA	N/A	N/A
Multigram	WA	WA	N/A	N/A	N/A
Birds on a Wire	WA	WA	WA	N/A	N/A
Kolone	WA	WA	N/A	N/A	N/A

Continued on next page

Table B.3: Detailed results of Gemini without RAG (Continued)

Name	1st	2nd	3rd	4th	5th
Book Shelves	WA (2/52)	WA (2/52)	WA (2/52)	WA (2/52)	WA (2/52)
Eating Out	WA (6/21)	WA (2/21)	WA (2/21)	N/A	N/A
House of Cards	WA	WA	N/A	N/A	N/A
Arctic Network	RTE	RTE	RTE	N/A	N/A
Where to Live?	TLE (1/5)	TLE (1/5)	N/A	N/A	N/A
Diagonal Cut	WA	WA	N/A	N/A	N/A
Adding Words	RTE	RTE	RTE	RTE	N/A
10 Kinds of People	WA	WA	WA	WA	RTE
A1 Paper	RTE	WA	WA	N/A	N/A
Alphabet Animals	WA	WA	WA	WA	WA
Euclid's Game	TLE	TLE	WA	N/A	N/A
Freckles	RTE	RTE	RTE	RTE	N/A
Cakey McCakeFace	TLE (4/15)	WA	WA	N/A	N/A
Game Night	WA	WA	N/A	N/A	N/A
Conservation	WA	WA	WA	WA	N/A
Wolf	RTE	WA	WA	N/A	N/A
Thesaurus	WA	WA	WA	N/A	N/A
Trending Topic	RTE	RTE	WA	RTE	N/A
Unique Dice	WA (1/44)	WA (1/44)	WA (1/44)	N/A	N/A
Sensor Network	RTE	RTE	RTE	N/A	N/A
Antenna Analysis	WA	WA	N/A	N/A	N/A

Continued on next page

Table B.3: Detailed results of Gemini without RAG (Continued)

Name	1st	2nd	3rd	4th	5th
Unown Code (Easy)	WA	WA	N/A	N/A	N/A
Stol	WA	WA	WA	N/A	N/A
Wedding	RTE	RTE	RTE	RTE	RTE
Abridged Reading	WA	WA	WA	N/A	N/A
Lemonade Trade	WA	WA	WA	WA	N/A
Wipe Your Whiteboards	RTE	WA	WA	N/A	N/A
0-1 Sequences	WA (2/41)	WA	WA	WA	N/A
Bag of Tiles	WA	RTE	RTE	RTE	N/A
Taxi Cab Scheme	RTE	RTE	RTE	N/A	N/A
Global Warming	WA (1/57)	WA (1/57)	WA (1/57)	N/A	N/A
Biggest Slice	RTE	RTE	RTE	RTE	N/A
EvenOdd	RTE	WA	N/A	N/A	N/A
Max Arithmetic Subsequence	WA	WA	WA	WA	WA
Number Sets (Hard)	RTE	WA	RTE	N/A	N/A
Radar	WA	RTE	RTE	RTE	RTE
String Multimatching	WA	WA	WA	WA	WA
Colliding Traffic	RTE	RTE	WA	CE	WA
Sumsets	WA (2/33)	WA	WA (3/33)	WA (2/33)	WA
Equations	RTE	RTE	RTE	RTE	N/A
Dvoniz	WA	WA	CE	N/A	N/A

Continued on next page

Table B.3: Detailed results of Gemini without RAG (Continued)

Name	1st	2nd	3rd	4th	5th
Liga	RTE	RTE	WA	RTE	RTE
Money Transfers	RTE	RTE	RTE	WA	WA
Product Divisors	WA	WA	WA	WA	N/A
Partial Linear Equation Solver	RTE	RTE	RTE	RTE	RTE
Binary search tree	WA	WA	WA	WA	N/A
Gas Station Numbers	WA	WA	WA	WA	N/A
Lone Rook	WA	WA	WA	WA	CE
Cycles (Hard)	RTE	WA	TLE	N/A	N/A
Icons in the Toolbar	WA	WA	N/A	N/A	N/A
Division	RTE	RTE	RTE	RTE	RTE
Endless Knight	RTE	WA	RTE	RTE	N/A
GCD Sum	WA	WA	WA	WA	N/A
Guess the Numbers	RTE	RTE	RTE	N/A	N/A
Basic	WA	WA	WA	WA	N/A
3-Sided Dice	RTE	RTE	CE	CE	WA
Crusaders of the Lost Mark	WA	WA	WA	WA	WA
Kolkrabbaleikarnir	WA (3/154)	WA (3/154)	WA (3/154)	WA (4/154)	N/A
1-D Frogger (Hard)	RTE	RTE	TLE (2/12)	TLE (2/12)	TLE (2/12)
Guess the Digits	WA	WA	WA	RTE	WA
Mravi	CE	CE	CE	CE	CE

Continued on next page

Table B.3: Detailed results of Gemini without RAG (Continued)

Name	1st	2nd	3rd	4th	5th
Closeness Queries	RTE	WA	WA	RTE	N/A
Frumtölutalning	RTE	RTE	RTE	RTE	RTE
A Different List Game	WA	WA	WA	WA	WA
Magical Mystery Knight's Tour	RTE	RTE	RTE	WA	WA
Towers of Powers 2: Power Harder	WA (1/31)	WA (1/31)	WA (1/31)	RTE	RTE
Uuu	RTE	RTE	RTE	N/A	N/A

B.4 Gemini With RAG Results

Table B.4: Detailed results of Gemini with RAG

Name	1st	2nd	3rd	4th	5th
Quadrant Selection	AC	N/A	N/A	N/A	N/A
Add Two Numbers	AC	N/A	N/A	N/A	N/A
Autori	AC	N/A	N/A	N/A	N/A
R2	AC	N/A	N/A	N/A	N/A
Bitte ein Bit	RTE	N/A	N/A	N/A	N/A
Two-sum	AC	N/A	N/A	N/A	N/A
Blandað Best	AC	N/A	N/A	N/A	N/A
Betting	RTE	N/A	N/A	N/A	N/A
Chanukah Challenge	RTE	N/A	N/A	N/A	N/A

Continued on next page

Table B.4: Detailed results of Gemini with RAG (Continued)

Name	1st	2nd	3rd	4th	5th
Solving for Carrots	AC	N/A	N/A	N/A	N/A
Reversed Binary Numbers	AC	N/A	N/A	N/A	N/A
Travelling Salesperson 2D	AC	N/A	N/A	N/A	N/A
Karte	RTE	N/A	N/A	N/A	N/A
Aaah!	RTE	RTE	N/A	N/A	N/A
Sjecista	WA (1/13)	N/A	N/A	N/A	N/A
Moscow Dream	AC	N/A	N/A	N/A	N/A
Exactly Electrical	WA	WA (1/17)	N/A	N/A	N/A
Hot Hike	AC	N/A	N/A	N/A	N/A
Above Average	RTE	RTE	N/A	N/A	N/A
Army Strength (Easy)	AC	N/A	N/A	N/A	N/A
Soft Passwords	WA (2/38)	WA (1/38)	N/A	N/A	N/A
Math Homework	WA	WA	N/A	N/A	N/A
Akcija	AC	N/A	N/A	N/A	N/A
Bazen	WA (1/13)	WA (1/13)	N/A	N/A	N/A
The Weight Of Words	WA	WA	TLE (8/23)	N/A	N/A
Closest Sums	RTE	WA	WA	N/A	N/A
ABC	RTE	WA (1/12)	N/A	N/A	N/A
3D Printed Statues	AC	N/A	N/A	N/A	N/A

Continued on next page

Table B.4: Detailed results of Gemini with RAG (Continued)

Name	1st	2nd	3rd	4th	5th
Supercomputer	TLE (3/13)	WA	WA	N/A	N/A
Baloni	WA	WA	WA	N/A	N/A
Multigram	WA (1/13)	WA	N/A	N/A	N/A
Birds on a Wire	RTE (1/11)	WA (1/11)	WA	N/A	N/A
Kolone	WA	WA	N/A	N/A	N/A
Book Shelves	WA (2/52)	WA (1/52)	WA	WA (2/52)	WA (3/52)
Eating Out	AC	N/A	N/A	N/A	N/A
House of Cards	WA	WA	N/A	N/A	N/A
Arctic Network	RTE	RTE	RTE	N/A	N/A
Where to Live?	RTE	RTE	N/A	N/A	N/A
Diagonal Cut	WA	WA	N/A	N/A	N/A
Adding Words	WA	RTE	WA	RTE	N/A
10 Kinds of People	RTE	WA	RTE	RTE	WA (1/25)
A1 Paper	WA	WA	WA	N/A	N/A
Alphabet Animals	WA (2/35)	WA (2/35)	WA (2/35)	WA (2/35)	WA (2/35)
Euclid's Game	RTE (1/2)	WA	WA	N/A	N/A
Freckles	WA	WA	RTE	RTE	N/A
Cakey McCakeFace	AC	N/A	N/A	N/A	N/A
Game Night	RTE	RTE	N/A	N/A	N/A
Conservation	RTE	WA	WA	WA (1/6)	N/A

Continued on next page

Table B.4: Detailed results of Gemini with RAG (Continued)

Name	1st	2nd	3rd	4th	5th
Wolf	RTE	RTE	RTE	N/A	N/A
Thesaurus	WA	RTE	RTE	N/A	N/A
Trending Topic	WA	WA	WA	WA	N/A
Unique Dice	WA	WA	WA	N/A	N/A
Sensor Network	RTE	RTE	RTE	N/A	N/A
Antenna Analysis	WA	WA	N/A	N/A	N/A
Unown Code (Easy)	WA	WA	N/A	N/A	N/A
Stol	WA	WA	WA	N/A	N/A
Wedding	RTE	RTE	RTE	RTE	RTE
Abridged Reading	RTE	RTE	RTE	N/A	N/A
Lemonade Trade	RTE	RTE	RTE	RTE	N/A
Wipe Your Whiteboards	WA	WA	RTE	N/A	N/A
0-1 Sequences	AC	N/A	N/A	N/A	N/A
Bag of Tiles	WA	WA	WA	RTE	N/A
Taxi Cab Scheme	WA	RTE	RTE	N/A	N/A
Global Warming	WA (1/57)	WA (1/57)	WA (1/57)	N/A	N/A
Biggest Slice	WA	WA	WA	WA	N/A
EvenOdd	WA (1/17)	TLE (2/17)	N/A	N/A	N/A
Max Arithmetic Subsequence	WA	RTE	RTE	RTE	RTE
Number Sets (Hard)	RTE	RTE	TLE (1/5)	N/A	N/A
Radar	WA	RTE	RTE	RTE	RTE

Continued on next page

Table B.4: Detailed results of Gemini with RAG (Continued)

Name	1st	2nd	3rd	4th	5th
String Multimatching	WA	RTE	RTE	RTE	RTE
Colliding Traffic	WA	RTE	RTE	RTE	RTE
Sumsets	WA (2/33)	WA (2/33)	RTE (1/33)	WA (1/33)	RTE (1/33)
Equations	RTE	WA	WA	RTE	N/A
Dvoniz	WA	WA	WA	N/A	N/A
Liga	WA	WA	WA	WA	WA
Money Transfers	WA	WA	RTE	RTE	RTE
Product Divisors	WA (1/16)	WA (1/16)	WA (1/16)	WA (1/16)	N/A
Partial Linear Equation Solver	RTE	RTE	WA	WA	RTE
Binary search tree	WA	WA	WA	WA	N/A
Gas Station Numbers	RTE	RTE	RTE	RTE	N/A
Lone Rook	WA	RTE	RTE	CE	RTE
Cycles (Hard)	WA	WA	WA	N/A	N/A
Icons in the Toolbar	WA	WA	N/A	N/A	N/A
Division	RTE	RTE	CE	RTE	RTE
Endless Knight	MLE (1/3)	WA	WA	WA	N/A
GCD Sum	WA	WA	WA	WA	N/A
Guess the Numbers	RTE	RTE	WA	N/A	N/A
Basic	CE	RTE	RTE	RTE	N/A
3-Sided Dice	WA	WA	RTE	RTE	RTE

Continued on next page

Table B.4: Detailed results of Gemini with RAG (Continued)

Name	1st	2nd	3rd	4th	5th
Crusaders of the Lost Mark	TLE	TLE	TLE	TLE	TLE
Kolkrabbaleikarnir	RTE (3/154)	WA (3/154)	WA (3/154)	WA (7/154)	N/A
1-D Frogger (Hard)	WA	RTE	WA	RTE	RTE
Guess the Digits	WA	WA	WA	WA	WA
Mravi	WA	WA	RTE	RTE	RTE
Closeness Queries	WA	WA	WA	WA	N/A
Frumtölutalning	WA	WA (13/124)	WA (94/124)	WA	WA (13/124)
A Different List Game	WA	WA	WA	WA	WA
Magical Mystery Knight's Tour	RTE	RTE	RTE	WA	WA
Towers of Powers 2: Power Harder	RTE	RTE	RTE	RTE	RTE
Uuu	RTE	WA (1/2)	WA	N/A	N/A

B.5 Code Generated by ChatGPT and Gemini

The final code generated for each attempt by both ChatGPT and Gemini is available in a public GitHub repository. The Link for the repository is available below. The structure of the repository is as follows. The root of the repository consists of 100 folders for each of the 100 assignments that were listed in Appendix A. Inside each folder, there are another two folders representing ChatGPT and Gemini, respectively. These folders contain two Python code files named after the method that was used to generate them (utilizing RAG and not utilizing it). Figure B.2 illustrates the file structure

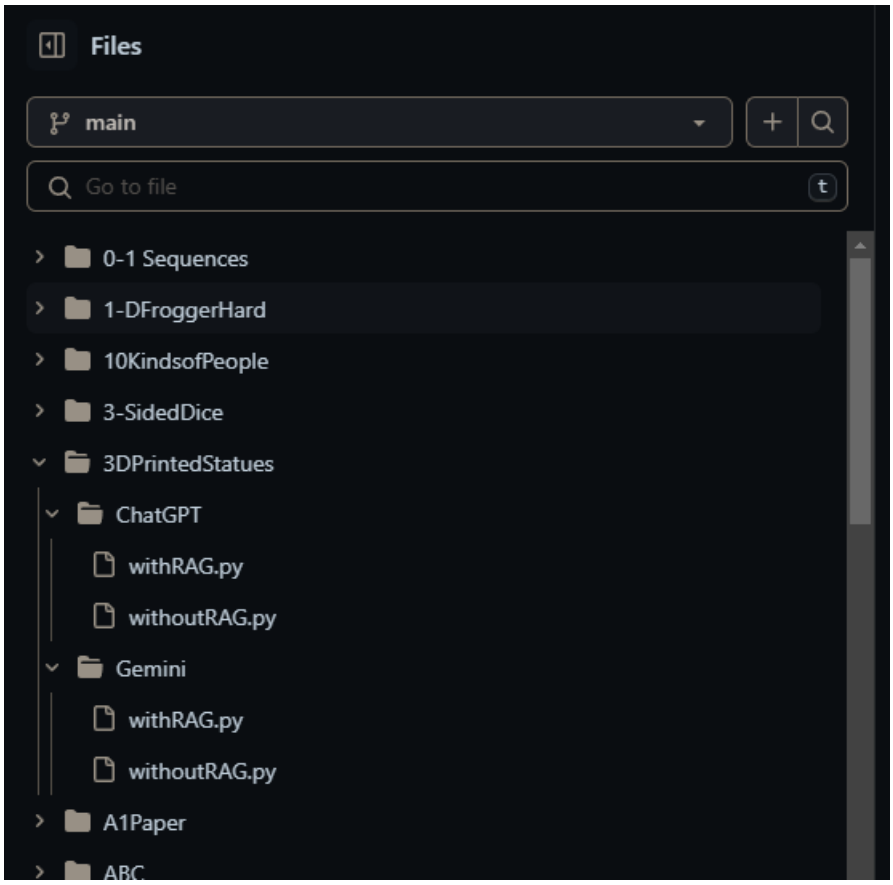

Figure B.2: Illustration of the file structure in the GitHub repository.

of the repository.

Link to the repository: https://github.com/AliAsbai/Thesis-Experiment-2024/

TRITA-EECS-EX-2024:449
Stockholm, Sweden 2024

www.kth.se