LUNDS UNIVERSITET
Ekonomihögskolan

Institutionen för informatik

Informatikstudenters användning av ChatGPT inom programmering

En kvalitativ studie om hur informatikstudenter använder ChatGPT och promptar

Kandidatuppsats 15 hp, kurs SYSK16 i Informatik.

Författare: Jonatan Carlsson
Jesper Hultkrantz

Handledare: Blerim Emruli

Rättande lärare: Umberto Fiaccadori
Paul Pierce

Informatikstudenters användning av ChatGPT inom programmering : En kvalitativ studie om hur informatikstudenter använder ChatGPT och promptar

ENGELSK TITEL:Informatics students' use of ChatGPT in programming : A qualitative study of how informatics students use ChatGPT and prompts

FÖRFATTARE: Jonatan Carlsson, Jesper Hultkrantz

UTGIVARE: Institutionen för informatik, Ekonomihögskolan, Lunds universitet

EXAMINATOR: Osama Mansour, Docent

FRAMLAGD: maj, 2024

DOKUMENTTYP: Kandidatuppsats

ANTAL SIDOR: 45

NYCKELORD: Stora språkmodeller, Prompt engineering, ChatGPT, Programmering

SAMMANFATTNING (MAX. 200 ORD):
Denna studie utforskar hur informatikstudenter använder sig av AI-verktyget ChatGPT vid programmering och hur de använder sig av prompt engineering för att effektivt utnyttja verktyget. Genom kvalitativa intervjuer med studenter samlar arbetet insikter kring deras upplevelser och strategier när de använder ChatGPT för att lösa programmeringsproblem. Resultatet visar på att studenterna använder ChatGPT hela tiden när de programmerar, speciellt när de felsöker och när de försöker lära sig något nytt. Resultatet visar också på att studenterna anser att kvalitén på deras prompts påverkar användbarheten av svaren från ChatGPT. Studenterna visar också tendenser på att använda etablerade prompttekniker, speciellt Chain-of-thought.

Figurer

Tabeller

1 Introduktion

1.1 Bakgrund

Sedan OpenAI släppte deras LLM (Large-Language Model) ChatGPT 3 i december 2022 har hela världen varit uppslukad av AI. Idag, knappt två år senare, används dessa modeller inom de flesta områden. Modellerna assisterar människor vardagligen med att generera texter, förenkla svårhanterlig information och hjälper till att lösa kodningsproblem (Gartner, 2023). I en nära framtid kommer LLMs användas i företag för att öka produktiviteten och sänka kostnader menar Gartner (2023). Utvecklare som använder LLMs kan klara av programmeringsuppgifter avsevärt snabbare än de som kodar själva enligt Githubs (Kalliamvakou, 2022) experiment. I experimentet fick hälften av utvecklarna använda Github Copilot och den andra hälften fick klara sig utan. Resultatet av detta blev att gruppen med Copilot blev klara 55% snabbare och lyckades med uppgiften i högre grad (78% mot 70%). Dessa AI-verktyg har funnits under en mycket kort tid vilket betyder att det finns begränsad forskning om detta och Github (Kalliamvakou, 2022) menar att det är svårt att definiera en utvecklares produktivitet. Utöver snabbhet hjälper Copilot utvecklare att öka produktiviteten genom att spara på utvecklarens mentala kraft och avlasta utvecklaren med repetitivt kodningsarbete (Kalliamvakou, 2022).

Med framstegen inom artificiell intelligens har modeller såsom ChatGPT 3.5 och 4 visat på betydande utvecklingar i prestanda. Dessa förbättringar drivs av avancerade algoritmer och omfattande mängder träningsdata, vilket har lett till en gradvis förstärkning av modellernas kapacitet över tid. Specifikt, ChatGPT 4 har demonstrerat en ökad förmåga att generera text av hög kvalitet och en förbättrad förståelse för en bredare variation av ämnen. Enligt en teknisk rapport från OpenAI (2023) har jämförande tester mellan ChatGPT 3.5 och ChatGPT 4 visat att den senare modellen presterar bättre i flertalet testfall. Denna förbättrade funktionalitet har potential att bredda modellernas användningsområden, inklusive skapandet av marknadsföringsmaterial samt förmågan att känna igen och analysera objekt i bilder. Dessutom har ChatGPT 3 fortsatt att utvecklas för att bättre svara mot de globala användarnas skiftande behov. De snabba förbättringarna av dessa modeller understryker den pågående utvecklingen inom fältet och dess potentiella påverkan på framtida applikationer inom olika yrkesfält.

Stora språkmodeller (Large Language Models, LLMs) har betydande potential, men deras effektivitet beror på användarnas förmåga att formulera frågeställningar, ett fenomen känt som "prompt engineering". "Prompt engineering" är en teknik som syftar till att förbättra LLMs output genom noggrant formulerade instruktioner. Enligt Sahoo et al. (2024), finns det ingen etablerad metod för att fullt ut automatisera och spåra effekten av olika prompts, vilket delvis beror på områdets nyhet (Rodriguez et al. 2023; Sahoo et al. 2024). Forskning på området antyder att "prompt engineering" kan vara ett effektivt sätt att utnyttja LLMs anpassningsbara natur. Flera prompttekniker har utvecklats för att förbättra output i programmeringsspecifika sammanhang, inklusive "Scratchpad prompting", "Program of thought", och "Chain-of-Thought" (Sahoo et al. 2024).

1.2 Problemområde

ChatGPT har blivit ett verktyg som allt mer och mer används av studenter för att hämta information. AI-verktyg för inlärning används generellt i allt större utsträckning (Wang et al. 2024), speciellt inom området utbildning (Velásquez-Henao et al. 2023). Dess förmåga att hantera naturlig språkbearbetning är väldigt kraftfull, vilket reflekterar potentialen den har för att maximera studenters inlärning. Tidigare forskning om hur ChatGPT används av studenter har inte haft sitt fokus på prompt engineering och hur man som student kan samla information på det mest effektiva sättet (Wang et al. 2024). En prompt är den text användaren förser ChatGPT med och utifrån detta genereras ett svar (Velásquez-Henao et al. 2023). En prompt i sig beskrivs som själva kommunikationen mellan ChatGPT och användaren. Wang et al. (2024) beskriver begreppet prompt engineering som en konst. Begreppet syftar till att bygga upp en prompt som på bästa möjliga sätt ska ge användaren ett svar vars innehåll matchar det som efterfrågas (Wang et al. 2024).

Problemet med att skriva en prompt, vars genererade svar är just det som efterfrågas, ligger i bristande kunskap hos användaren och svårigheten med att formulera en effektiv prompt. Speciellt när användaren önskar få svar på något väldigt specifikt där det krävs att prompt:n även måste förses med specifik information. En prompt med otydliga avgränsningar leder i sin tur till ett otydligt svar (Velásquez-Henao et al. 2023). OpenAI Platform (n.d.) ger förslag på hur en dålig respektive bra prompt kan se ut. Den dåliga prompt:n är enligt följande citat ”Write code to calculate the Fibonacci sequence.” och den bra ”Write a TypeScript function to efficiently calculate the Fibonacci sequence. Comment the code liberally to explain what each piece does and why it's written that way.”. Det är viktigt att alla detaljer som rör det som efterfrågas förses för att få ett svar vars innehåll är relevant (OpenAI Platform, n.d.). Cain (2024) menar på att en prompt är i direkt relation till en LLM:s kapacitet. För att kunna utnyttja denna kapacitet behövs avsevärd kunskap inom området prompt engineering (Cain, 2024).

I och med bristen av tidigare forskning om studenters lärande, genom att hämta information från ChatGPT med fokus på prompt engineering, har den här uppsatsens fokus vuxit fram. För att en student på bästa sätt i sitt lärande ska kunna utnyttja den kapacitet som ChatGPT besitter, är kunskap inom prompt engineering av väldigt stor vikt. AI verktyg såsom ChatGPT används allt mer av studenter i sitt eget lärande och att besitta kunskap inom prompt engineering kan ge dem ett övertag kunskapsmässigt under deras utbildning jämfört med andra studenter. Det kommer i framtiden bli allt mer viktigt att besitta denna kunskap eftersom AI-verktygs fortsatta användning ökar och blir en större del av studentens egna lärande.

Genom att undersöka hur Informatik-studenter löser olika programmeringsproblem med hjälp av ChatGPT och hur de integrerar prompt engineering kan den här studien bidra med värdefulla insikter. Universitet och skolor generellt kan använda sig av resultaten från den här studien för att förstå vilken vikt prompt engineering har bland studenter i sitt eget lärande och därpå överväga att inkludera det som en del av skolans kursutbud. Detta för att ge studenter samma utgångsläge i användandet av LLM:s som exempelvis ChatGPT.

1.3 Forskningsfråga

Problembeskrivningen leder till denna fråga:

- Hur använder informatikstudenter AI-chatbots såsom ChatGPT för att lösa programmeringsproblem, och på vilket sätt integrerar de prompt engineering i denna process?

1.4 Syfte

Under problemområdet beskrivs begreppet prompt engineering som väldigt viktigt i relation till hur väl en LLM kan utnyttjas. Det finns en brist bland tidigare forskning som undersöker kunskapen om prompt engineering bland studenter och hur de kan utnyttja det i sitt eget lärande när de använder exempelvis en LLM såsom ChatGPT. Denna brist av tidigare forskning har således format syftet för den här studien, att undersöka hur ChatGPT används av studenter som ett verktyg inom programmering samt hur de integrerar prompt engineering i processen.

1.5 Avgränsningar

Studiens fokus kommer att ligga på vilken effektivitet prompt engineering har för informatikstudenter när de löser programmeringsproblem med hjälp av AI-verktyg. I det här fallet kommer enbart ett verktyg/LLM, ChatGPT, användas och studeras för att genomföra studien med avseende på intervjuerna och litteraturgenomgången. Studien kommer endast att inkludera studenter som studerar på det systemvetenskapliga kandidatprogrammet i Lund.

2 Litteraturgenomgång

2.1 Stora språkmodeller

2.1.1 ChatGPT

ChatGPT är en stor språkmodell skapad av företaget OpenAI. Med hjälp av AI kan de göra en stor mängd olika uppgifter (Hughes, 2023). Modellen kan skapa mänskligt naturligt språk, översättning av språk, känna igen bilder och generera datakod i många olika programmeringsspråk. ChatGPT är en av de mest starka och kraftfulla stora språkmodellerna i världen, den tillhandahåller 175 miljarder parametrar (Hughes, 2023). Parametrarna är de som skapar förmågan hos en stor språkmodell att förstå sig på och generera naturligt språk.

Stora språkmodeller har enorma möjligheter och med deras förmåga att skapa naturligt språk kan de verka i många olika områden som till exempel dialogsystem, textsammanfattning och maskinöversättning OpenAI (2023). Samtidigt menar OpenAI (2023) att språkmodellerna har sina begränsningar, till exempel kan deras senaste modell GPT-4 inte lära sig från tidigare upplevelser och kan bara ta in en begränsad mängd context i beaktning i sin problemlösning.

2.1.2 Stora språkmodeller som verktyg inom programmering

Enligt Gartner (2024) kommer 75 procent av alla mjukvaruutvecklare inom företag att använda AI-baserade kodassistenter senast 2028. Denna trend understryker hur viktiga språkmodeller och AI-verktyg blir i programmeringsmiljöer. Stor efterfrågan på dessa verktyg beror på deras förmåga att effektivisera kodningsprocessen, förbättra kodkvaliteten och minska utvecklingstiden. Det framhåller även vikten av att integrera och anpassa dessa teknologier för att stödja utvecklarnas behov, vilket ytterligare driver innovation och effektivitet inom mjukvaruutveckling.

I dagsläget kan AI användas som verktyg på flera sätt inom kodning, där stora språkmodeller erbjuder hjälp genom att automatisera delar av programmeringsprocessen. Enligt Leung och Murphy (2023), kan dessa modeller utvidga deklarativt minne hos utvecklare genom integrerade utvecklingsmiljöer, förbättra produktionseffektivitet genom att hantera kodtransformationsuppgifter, och stödja arbetsminnet genom att koordinera mellan olika kognitiva processer i mjukvaruutvecklingen. Dessa assistenter kan avsevärt öka en utvecklares kognitiva kapacitet genom att minska mentala overhead och effektivisera informationshanteringen.

2.2 ChatGPT som ett verktyg

ChatGPT:s popularitet har spridit sig till många sektorer, en av dessa är inom utbildning (Das & J.V, 2024). En av orsakerna till verktygets uppskattning bland studenter ligger delvis i dess enkla användarvänlighet där det inte ställs några krav på någon teknisk expertis för att kunna använda det. Användningen av verktyget ställer endast krav hos användaren att kunna skriva

mänskligt språk. Detta gör det möjligt för studenter, oberoende av vilken akademisk nivå de besitter, att kunna använda och utnyttja vad verktyget har att erbjuda. Att bruka ChatGPT liknar till viss del hur en student hade sökt information via en sökmotor på internet, där man söker med hjälp av olika fraser och nyckelord (Cain, 2024).

2.2.1 Användningsområden

Både lärare och studenter använder sig idag av detta AI-verktyg (Das & J.V, 2024), men med avseende på den här uppsatsen kommer fokuset ligga på studenters användning av ChatGPT. Studenter använder ChatGPT för exempelvis förberedelse inför ett prov, lösa problem, förtydligande av koncept och analysera data (Das & J.V, 2024). Cain (2024) menar på att ett AI-verktyg såsom ChatGPT kan förse och specialanpassa inlärning så den ska stämma överens med de specifika behov en student kan tänkas ha. Das & J.V. (2024) beskriver ytterligare fördelaktiga egenskaper som studenter får ut av när de använder ChatGPT, dessa är bland annat att de får feedback, får hjälp med ide-skapande samt att det blir tidsbesparande för dem.

2.2.2 Utmaningar

En stor utmaning som studenter möter i sitt användande av ChatGPT är deras förmåga att formulera bra inputs/prompts, den text som skickas till verktyget. Cain (2024) menar på att en stor del av att kunna utnyttja ett AI-verktyg såsom ChatGPT till fullo ligger i själva texten, också kallat en prompt, som användaren skriver och förser verktyget med. Att skriva en effektiv prompt hör ihop med begreppet ”Prompt Engineering”, för att förstå vad begreppet innebär mer utförligt se punkt 2.3.

En välformulerad prompt är i direkt relation till den respons verktyget ger. Med en effektiv och välformulerad prompt kan användaren få respons på just det som efterfrågas. En av den största utmaningen när det kommer till att formulera en effektiv prompt ligger i förmågan att hitta en balans mellan hur komplex och enkel en prompt ska vara. En obalans mellan dessa två kan exempelvis leda till en respons som tolkar en prompt fel (Cain, 2024). Velásquez-Henao et al. (2023) beskriver ytterligare en stor utmaning där en prompt som är otydligt formulerad kan leda till att responsen innehåller hallucinationer. Det är alltså då av stor vikt att källgranska responsen och kontrollera att innehållet är korrekt (Velásquez-Henao et al. 2023).

Förståelsen och kunskapen inom ”Prompt Engineering” beskrivs enligt Wang et al. (2024) som ett måste när det kommer till att effektivisera sitt användande med ChatGPT. Relationen mellan vad användaren vill få ut av ChatGPT genom en prompt och vad ChatGPT förstår av det är essentiellt. Det är just där ”Prompt engineering” kommer in som ett verktyg för att möjliggöra den relationen (Wang et al. 2024).

2.2.3 ChatGPT som ett verktyg inom Systemutveckling

En student i rollen som systemutvecklare kan använda sig av ChatGPT som ett verktyg och hjälpmedel inom flera olika områden. Hutchinson (2023) nämner sex stycken sådana områden där den första är att som student kunna lära sig ett nytt programmeringsspråk. Studenter kan ställa frågor de har till ChatGPT om det nya språket de lär sig och kan därefter få vägledning

till hur koden fungerar. Det andra området som nämns är att optimera sin kod. Efter att ha försett ChatGPT med den kod som ska optimeras kan verktyget därefter förslagsvis föreslå kod som kan förbättras utifrån dess bedömning av koden.

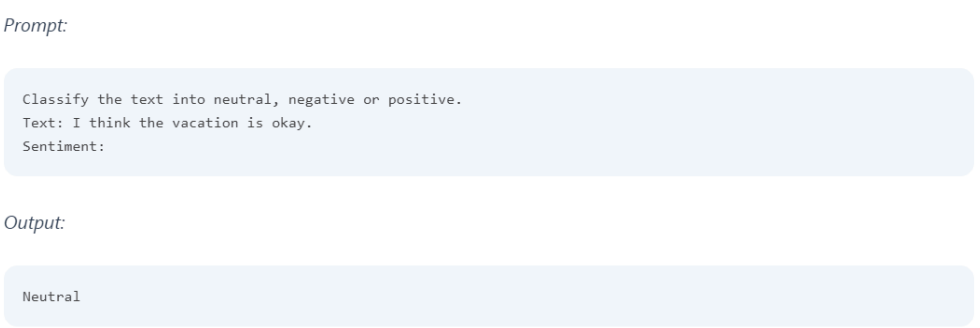
Det tredje området är ”Debugging”, verktyget kan här hjälpa studenten med att felsöka och peka ut fel i koden. Det fjärde området är ”Brainstorming”. ChatGPT kan här hjälpa och vägleda studenten till att komma på nya idéer för exempelvis ett nytt projekt. Det femte området är hur ChatGPT kan hjälpa utvecklare att komma på tester som ska testa funktionaliteten av deras projekt. Det sjätte och sista området är hur verktyget kan hjälpa studenter att överkomma problem de stöter på. Verktyget kan då hjälpa och föreslå tillvägagångssätt för att lösa problemet (Hutchinson, 2023). ChatGPT som ett verktyg för att bemöta de presenterade områdena är ett effektivt sätt för att öka sin effektivitet i en roll systemutvecklare.

2.3 Prompt Engineering

Prompt engineering är en metod inom naturlig språkbehandling för att skapa precisa instruktioner, eller "prompter", för språkmodeller. Genom att noggrant formulera frågor eller påståenden styr den modellens generering av svar. Det innefattar noggranna val av nyckelord, grammatisk struktur och kontextuella ledtrådar för att uppnå önskade resultat och optimera output från språkmodeller. Prompterna bör vara simpla, specifika och tydliga enligt Prompt Engineering Guide (2024) för att maximera effektiviteten. Utöver generella riktlinjer för prompting utforskar vi i denna litteraturgenomgång olika tekniker för att skriva prompts. Det är värt att nämna att det inte finns en promptteknik som är korrekt för alla problem utan det är problemet som styr vilken teknik man borde använda.

2.3.1 Zero-Shot Prompting

Large Language Models (LLMs) är framstående på att lära sig från exempel och generera relevant output baserat på dessa exempel. Inom prompting kallas dessa exempel ofta för "shots". Zero-shot prompting innebär att en prompt ges utan några exempel för att vägleda LLM. I Prompt Engineering Guide (2024) visas ett exempel på lyckad output från en Zero-shot prompt där GPT-3 kan identifiera om en mening är neutral, negativ eller positiv utan specifika exempel på sådana meningar. Trots att Zero-shot prompting är utmanande för LLMs, påpekar Prompt Engineering Guide (2024) att de klarar det väl tack vare deras förmåga att följa instruktioner och omfattande träning på stora dataset.



Figur 1: Prompt Engineering Guide: Prompt 1, (Prompt Engineering Guide, 2024)

Wei et al. (2022) påpekar att LLMs har svårigheter med Zero-shot prompting jämfört med Few-Shot prompting, särskilt när det gäller uppgifter som läsförståelse, svarsförmåga på frågor och "Natural Language Inference (NLI)". NLI handlar om hur man avgör kopplingen mellan flera meningar, vilket är en viktig del hur LLMs resonerar om text. Detta visar på vikten av att balansera förväntningar och att överväga olika promptingstrategier beroende på uppgiftens krav och LLMs kapacitet. Till exempel kan zero-shot prompting vara användbart i enkla uppgifter men svårare uppgifter får man bättre resultat genom att använda andra tekniker menar Prompt Engineering Guide (2024).

2.3.2 Few-Shot Prompting

Som vi har konstaterat i föregående stycke betyder termen “shot” exempel på problemet man vill att AI-modellen skall lösa. Detta betyder alltså att tekniken“Few-Shot Prompting" inkluderar ett antal exempel på problemet för att förklara kontexten till AI-modellen. Enligt Prompt Engineering Guide (2024) kan dessa exempel som inkluderas i prompten styra modellen till bättre prestanda. När man skapar en prompt med Few-Shot Prompting kan det vara fördelaktigt att använda sig av Min et al. (2022) riktlinjer. De trycker på att använda sig av etiketter och att använda korrekt format kan påverka AI-modellens prestanda. Figur X är ett exempel på hur man kan använda sig av dessa riktlinjer. Prompten följer en tydlig struktur: “En vanlig mening!” // Etikett. Detta ger exempel på positiva och negativa meningar, visar hur formatet skall vara med “//” och en etikett som säger antingen neutral, positiv eller negativ. I den sista meningen finns det inget etikett som påvisar vilken ton meningen har vilket antyder att AI-modellen skall lista ut det själv.

```
This is awesome! // Negative
This is bad! // Positive
Wow that movie was rad! // Positive
What a horrible show! //
```

Figur 2: Prompt Engineering Guide: Prompt 2, (Prompt Engineering Guide, 2024)

Även fast “Few-Shot Prompting” kan hjälpa LLMs lösa mer komplexa problem än “Zero-Shot Prompting” har tekniken sina limiteringar. Prompt Engineering Guide (2024) testade tekniken med ett experiment där de gav modellen sju stycken udda tal och ställde frågan om summan av talen blev ett positivt tal. För att få ett bättre resultat av LLMs kan det alltså vara nödvändigt att använda sig av ännu mer avancerade prompttekniker, som till exempel “Chain-of-thought”.

```
The odd numbers in this group add up to an even number: 4, 8, 9, 15, 12, 2, 1.
A: The answer is False.
The odd numbers in this group add up to an even number: 17,  10, 19, 4, 8, 12, 24.
A: The answer is True.
The odd numbers in this group add up to an even number: 16,  11, 14, 4, 8, 13, 24.
A: The answer is True.
The odd numbers in this group add up to an even number: 17,  9, 10, 12, 13, 4, 2.
A: The answer is False.
The odd numbers in this group add up to an even number: 15, 32, 5, 13, 82, 7, 1.
A:
```

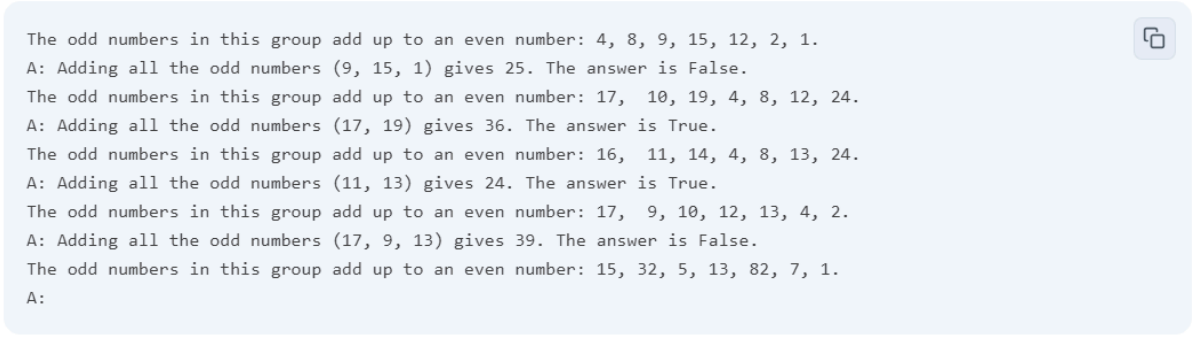
Figur 3: Prompt Engineering Guide: Prompt 3, (Prompt Engineering Guide, 2024)

2.3.3 Chain-of-thought

Att utvidga och förstärka Large Language Models har visserligen förbättrat deras prestanda och effektivitet när det gäller att generera resultat från begränsade exempel, men detta ensamt räcker inte för att göra LLMs skickliga på att lösa och resonera kring komplexa problem över olika områden. Enligt Wei et al. (2023) har ökningen av LLMs storlek och kraft visserligen medfört framsteg, men när det kommer till utmanande uppgifter som kräver djupare förståelse, såsom problemlösning och resonemang inom flera områden, har LLMs ännu inte nått önskad nivå av skicklighet. Detta betonar behovet av att utveckla ytterligare strategier och tekniker för att förbättra LLMs förmåga att hantera och resonera om komplexa problem, bortom bara ökad storlek och effektivitet. Wei et al. (2023) presenterade en lösning till detta, prompttekniken “Chain of Thought” (COT).

I Wei et al. (2023) presenteras tanken att ge språkmodeller förmågan att generera en sammanhängande kedja av tankar, liknande det sätt människor löser komplicerade resonemangsproblem som multi-stegs matematiska ordproblem. Denna kedja av tankar bryter ner problemet i mellansteg och löser varje steg innan den ger det slutgiltiga svaret. Genom att tillhandahålla exemplar av tankeskedjor för “Few-Shot prompting” kan tillräckligt stora språkmodeller generera liknande tankesekvenser. Denna metod har flera fördelar, inklusive förmågan att bryta ner flerstegsproblem och ge en tolkningsbar inblick i modellens beteende. Chain-of-thought resonemang är användbart för en rad uppgifter, inklusive matematiska ordproblem och vardagligt resonemang, och kan enkelt framkallas i stora språkmodeller genom att inkludera exempel på tankesekvenser i “Few-shot prompting” exemplar. Den empiriska undersökningen i Wei et al. (2023) illustrerar nyttan med chain-of-thought prompting för aritmetiska problem.

Prompt Engineering Guide (2024) testar denna prompttekniken genom att utöka sin prompt med de sju udda talen när de testade “Few-Shot Prompting”. I figur 4 lägger de denna gång till en utökning i sin prompt. De lägger till ett extra steg där de ger summan av de tre första udda talen. Detta ger AI-modellen en lösning, ett delproblem för att sedan fortsätta lösa problemet själv. Mycket riktigt fick Prompt Engineering Guide (2024) ett korrekt resultat av GPT-3 efter COT-prompten.



Figur 4: Prompt Engineering Guide: Prompt 4, (Prompt Engineering Guide, 2024)

Efter att ha testat flera olika stora språkmodeller på komplexa matematikproblem, upptäckte Wei et al (2023) att användningen av Chain-of-Thought resulterade i en betydande ökning av andelen lösta problem jämfört med andra metoder. Detta resultat stämmer dock bara för de större språkmodellerna med över 100 miljarder inparametrar. I Gemini Team Googles (2023) tekniska rapport på deras stora språkmodell Gemini får de liknande resultat. I deras

experiment där de använder sig av Chain-of-thought får de betydligt bättre prestanda, särskilt i sin mest avancerade modell Gemini Ultra.

2.4 Tidigare forskning inom ämnet

2.4.1 Motivering

De två studier som presenteras nedan kommer att användas för främst två saker. Att ge inspiration till utformningen av den här uppsatsens metod samt som ett underlag för att jämföra resultaten från den här uppsatsens studie med vad tidigare forskning kommit fram till. Det presenterade studierna är relevanta för den här uppsatsens studie då de undersöker, även om inte exakt samma tillvägagångssätt som den här uppsatsens metod har, vikten av prompt engineering bland studenter inom programmering, vilken kvalité svar från AI-verktyg har och hur olika nivåer av kunskap inom prompt engineering speglar detta. Detta kommer att ge oss ett underlag för att i senare del kunna diskutera och jämföra den här studiens resultat med vad de två presenterade studierna nedan kommit fram till.

2.4.2 Studie 1

I en studie utförd av Wang et al. (2024) vill de undersöka huruvida college-studenter påverkas av prompt engineering när de hämtar information via ChatGPT. Deras titel på studien är ”Unleashing ChatGPT’s Power: A Case Study on Optimizing Information Retrieval in Flipped Classrooms via Prompt Engineering”. De har tagit fram två stycken frågeställningar som ska kunna svara på det studien vill undersöka. Den första är ”Does mastering prompt engineering methods help improve the quality of information students obtain from ChatGPT?” och den andra ”How can the content of prompt engineering be effectively arranged in teaching to enhance the quality and efficiency of flipped classroom instruction?”. För att kunna svara på dessa frågeställningar använde de sig i studien utav collegestudenter, alla från samma klass under sitt fjärde år. Studenterna, 24 stycken, delades slumpmässigt in i två grupper, en experimentell och en kontroll.

Den experimentella gruppen fick ta del av kunskap inom prompt engineering medan kontrollgruppen inte fick ta del av den kunskapen. Till en början gavs de båda test-grupperna en programmeringsuppgift att lösa inom området TCP(transmission control protocol), studenterna hade inte någon tidigare erfarenhet inom det området. Uppgiften gick ut på att i python skriva ett program där en klient skickar en sträng till en server den har anslutit till. Efter att ha tagit del av uppgiften blev de ombedda att ställa ChatGPT fem stycken frågor för att försöka lösa uppgiften. Vidare fick de båda grupperna en utförlig genomgång inom TCP för att bli insatta i ämnet. Därefter blev studenterna igen ombedda att skriva fem stycken frågor till ChatGPT för att lösa uppgiften. I det avslutande steget fick experimentgruppen en genomgång av olika metoder inom prompt engineering som de sedan applicerade på fem ytterligare frågor som de skrev till ChatGPT för att lösa uppgiften. Ett ramverk som heter ´CRISPE´ introducerades för studenterna i experimentgruppen som de skulle använda sig av. Kontrollgruppen fick endast en enkel genomgång av några grundläggande koncept inom prompt engineering inför den sista omgången av frågor de ska skriva (Wang et al. 2024).

För att kunna analysera detta gjorde Wang et al. (2024) en insamling av alla svar som

ChatGPT gett till de frågor studenterna skrivit. Denna data används sedan för att kunna mäta svarens kvalitet. Efter att de genomfört sin analys fann de att experimentgruppen, vars frågor skrevs efter att de tog del av kunskap inom prompt engineering, fick svar från ChatGPT som bedömdes ha mycket högre kvalitet än de jämfört med kontrollgruppen, som inte hade någon kunskap om prompt engineering. Vidare fann de också att svaren som samlades in från frågorna före och efter en genomgång om TCP inte gav någon avsevärd stor skillnad på kvalitet. Men att besitta kunskap inom ämnet uppgiften berör samt att ha kunskap inom prompt engineering beskrivs ha stor inverkan på kvalitén av de svar ChatGPT förser (Wang et al. 2024).

2.4.3 Studie 2

Denny et al. (2023) i deras studie ” Conversing with Copilot: Exploring Prompt Engineering for Solving CS1 Problems Using Natural Language” undersöker hur effektiv Github Copilot är när det kommer till att lösa programmeringsproblem och hur prompt engineering kan användas som hjälpmedel. Studien utgår ifrån tre stycken frågeställningar ”How well does Copilot perform, at the current time, on a public repository of CS1 programming problems?”, ”To what extent do natural language modifications to the problem description lead to the generation of successful solutions after Copilot initially fails?” Och ”What commonalities exist amongst problems that Copilot cannot solve, despite reasonable prompt engineering effort?”.

För att kunna svara på dessa frågeställningar har Denny et al. (2023) i studien använt sig av en metod som går ut på att testa 166 programmeringsproblem som de har hämtat från en hemsida som heter ´CodeCheck´. Problemen berör programmeringsspråket python och är indelade i fyra kategorier, dessa är ´Branches´, ´Strings´, ´Lists´ och ´Two-Dimensional Arrays´. Dessa kategorier var sedan indelade i 23 ytterligare kategorier som t.ex. ´Finding Elements´, ´Comparing Strings´ och ´Finding Strings´.

```
1   # Given a list of integers and a value, return a list
2   # of all positions of the value in the list. Do not use
3   # the find or index methods.
4
5   def findAllPositions(arr, val): # O(n)
        positions = []
        for i in range(len(arr)):
            if arr[i] == val:
                positions.append(i)
        return positions
```

Figur 5: Programmeringsproblem, (Denny et al. 2023)

Initialt kopierades varje enskild beskrivning av programmerings-problemen och klistrades in i Visual studio code som har ´extension´ Github Copilot aktiverat, se figur 5. Sedan kopierades den kod som Copilot generade och klistrades in i ´CodeCheck´ hemsidan där koden kontrolleras för att se om den löser problemet. Problemet markerades som godkänt om ´CodeCheck´ accepterade lösningen. Om koden från Copilot inte lyckades lösa problemet användes prompt engineering för att förtydliga problembeskrivningen tills den genererade koden löste problemet. Ett förtydligande gällande deras användning av prompt engineering gavs. Denny et al. (2023) följde inte någon specifik strategi för de prompter som skrevs efter att Copilot inte löste problemet på första försöket, utan olika strategier testades för att ge en

del frihet i att hitta den prompt engineering teknik som fungerade bäst för det givna problemet.

Denny et al. (2023) fann efter deras genomförda studie att Copilot, utan någon prompt engineering applicerad, lyckades lösa 79 av 166 problem, vilket motsvarar 47,6%. Efter att prompt engineering blev applicerat på de 89 problem som inte blev lösta i första steget blev ytterligare 53 problem, vilket motsvarar 60,9%, lösta och 23 problem förblev olösta. Detta resultat beskriver betona vikten av att kunna förstå och använda sig av prompt engineering för att som student kunna lösa den typ av problem som studien undersöker. Denny et al. (2023) betonar också vikten av att som student kunna bryta ner ett problem till konkreta steg när de använder Copilot som ett verktyg inom problemlösning.

2.5 Undersökningsmodell

Tabell 1: Undersökningsmodell

Tema	Subtema	Litteratur
ChatGPT		(Hughes, 2023), (OpenAI, 2023), Gartner (2024), (Leung & Murphy 2023)
ChatGPT som ett verktyg	● Användningsområden ● Utmaningar ● ChatGPT som ett verktyg inom Systemutveckling	(Das & J.V, 2024), (Cain, 2024), (Velásquez-Henao et al. 2023), (Wang et al., 2024), (Hutchinson, 2023)
Prompt Engineering	● Zero-Shot Prompting ● Few-Shot Prompting ● Chain-of-thought	Prompt Engineering Guide (2024), (Wei et al., 2022), Gemini Team Google (2023), (Wei et al., 2023), (Min et al., 2022)
Generalizability		(Lee & Baskerville, 2003)

3 Metod

3.1 Metodval

Efter att ha genomfört en utförlig litteraturgenomgång och samlat på sig en stor kunskap om området prompt engineering var vi redo för insamling av empirisk data. Vår ansats var av kvalitativ natur genom intervjuer. Detta kommer att ge en tydlig inblick i hur prompt engineering används av studenter i kontext till programmeringsproblem och deras förståelser av vissa specifika prompttekniker. Respondenterna studerar det Systemvetenskapliga kandidatprogrammet på Lunds universitet där användning av stora språkmodeller som ChatGPT som ett verktyg till programmeringen spelar en central roll i kandidaten.

3.1.1 Kvalitativa studier

Valet av kvalitativ metod baserades på studiens syfte och frågeställning. För att kunna svara på hur informatikstudenter använder sig av prompt engineering i kontexten till ett programmeringsproblem och deras kunskap om prompt engineering var en kvalitativ ansats det rätta valet. En kvalitativ studie tillåter studien att gå in på djupet med intervjufrågor jämfört med en kvantitativ, datan ifrån en kvalitativ studie kan vara detaljerad och mäter mer än bara det som kan mätas i siffror, vilket är till stor användbarhet till studier inom informatik (Oates et al. 2022). Jacobsen (2017) pratar också om att kvalitativ metod kan ge detaljerad data men samtidigt trycker han på att metoden ska väljas efter tillgång till resurser, ibland kan resurserna inte vara tillräckliga till allt som forskarna vill studera.

En kvalitativ ansats är särskilt lämplig för att utforska hur informatikstudenter använder ChatGPT inom programmering, då metoden tillåter en djupgående analys av studenternas interaktioner och uppfattningar. Genom intervjuer och observationer kan detaljerade insikter om deras specifika användning samt deras förståelse av prompt engineering och upplevda utmaningar samlas in. Att begränsa studiens omfång till en enda institution gör det möjligt att utföra en noggrann undersökning inom existerande resursramar, vilket är avgörande när resurserna är begränsade.

3.2 Insamlingsmetod och litteratursökning

3.2.1 Litteratursökning

Databaserna LubSearch och Google Scholar har använts som sökverktyg för att hitta relevant teori. Även böcker har använts för att stödja den metod uppsatsen använder sig av. För att hitta relevant teori inleddes sökprocessen i LubSearch och Google Scholar genom användningen av ett antal olika sökord. Sökorden som använts har varit:

- Prompt engineering
- ChatGPT
- LLM

- Programming
- Chain of Thought
- Natural Language
- Generalizability
- Comparison
- Students

Dessa ord har kombinerats på flera olika sätt för att försöka hitta den mest relevanta teorin för det ämnet uppsatsen berör. Sökningen har för det mesta skett på engelska för att ge ett bredare sökresultat. Det första steget efter att en sökning gjorts, i någon av de två databaser som använts, var att bedöma om sökresultatet var relevant. Detta gjordes genom att läsa igenom sammanfattningen för att snabbt få en inblick i vad materialet handlar om. Om det ansågs vara relevant lästes sedan hela materialet för att få en djupare insikt av vad det handlar om samt hitta delar som den här uppsatsen kan ha användning av.

För att säkerställa att teorin som samlats in är trovärdig har följande steg beaktats. De böcker som använts för att stödja uppsatsens metod har varit från trovärdiga utgivare samt att de använts som kurslitteratur. De publikationer tagna från LubSearch har alla varit ´Peer-Reviewed´ vilket säkerställer att de har akademisk kvalitet.

3.2.2 Urval av respondenter

I denna studie har urvalskriterierna för respondenterna grundats på tre specifika faktorer: För det första måste respondenten vara inskriven som student inom det systemvetenskapliga programmet. För det andra ska studenten ha påbörjat åtminstone sin fjärde termin inom programmet. Slutligen krävs att studenten besitter grundläggande färdigheter inom programmering. Initial kontakt med potentiella respondenter etablerades genom chattforum som innehöll grupper av systemvetare (till exempel Facebook grupper) samt e-postutskick. De studenter som uttryckte intresse för att delta i studien bjöds in till en personlig intervju, vilken arrangerades på plats i skolans lokaler. Således använde vi oss av en typ av bekvämlighetsurval eftersom att respondenterna finns inom vår egen utbildning. Bekvämlighetsurval är enligt Oates et al. (2022) ett urval baserat på forskarnas bekvämlighet att nå respondenterna i studien. I denna studiens fall passar detta urvalet bra då respondenterna är inom samma fakultet.

Utöver de nämnda kriterierna var det av intresse för studien att deltagarna hade praktisk erfarenhet av att använda stora språkmodeller, såsom ChatGPT. Detta för att kunna analysera deras insikter och förståelse av komplexa teknologier i en akademisk kontext. Studien syftade till att utforska hur väl dessa erfarenheter korrelerar med deras akademiska prestationer och tekniska kompetens.

Om det efterfrågades säkerställdes respondenternas anonymitet och integritet genom hela processen, från rekrytering till publicering av studiens resultat, i enlighet med gällande etiska riktlinjer för forskning. Samtliga respondenter visas enligt tabellen nedan.

Tabell 2: Respondenter

Respondent	Roll
R1	Informatikstudent
R2	Informatikstudent
R3	Informatikstudent
R4	Informatikstudent

3.2.3 Intervjuguide: Motivering av intervjufrågor

Jacobsen (2017) beskriver olika nivåer en intervju kan vara strukturerad, den kan exempelvis vara helt sluten men frågor som bara har fasta svarsalternativ eller helt öppen där en form av intervjuguide inte följs. Jacobson motiverar valet av strukturering för en intervju som inte är helt eller inte helt sluten, där en intervju inte har någon intervjuguide respektive frågor där svarsalternativen är fasta. Jacobsen (2017) menar på att en intervju ska följa vissa specifika teman som önskas tas upp och få svar på under intervjun. Genom att inkludera specifika teman i sin intervjuguide blir det som en garanti att de behandlas under intervjun.

Utifrån vad Jacobsen beskriver om strukturering för en intervju har en intervjuguide (se tabell 3) tagits fram med ett antal olika teman som ska tas upp samt frågor som ska ställas under respektive tema. Den teori som har gåtts igenom under punkt 2 (Litteraturgenomgång) har lagt grund för de teman som återfinns i intervjuguiden. Forskningsfrågan har spelat en roll i utformningen av frågorna för att säkerställa att intervjun ger svar på det som undersöks. Frågorna som återfinns i intervjuguiden ska därav främst undersöka hur respondenterna använder ChatGPT samt prompt tekniker för programmeringsproblem. Utöver de frågor som direkt berör forskningsfrågan har ett antal frågor, som berör bland annat bakgrund, upplevelse och kännedom, tagits fram som ska komplettera dessa. Tanken med de kompletterande frågorna är att deras svar ska ge mer förståelse till varför respondenterna svarar som de gör.

Frågorna i intervjuguiden har en tänkt fast ordningsföljd men med en öppenhet till att ändras om själva intervjun resulterar i att frågorna och och teman ändrar ordning. Intervjun kommer huvudsakligen att utgå utifrån de frågor som tagits fram i intervjuguiden, för att säkerställa att vissa teman berörs, men nya frågor samt följdfrågor kan komma att ställas. Detta ger respondenten mer frihet att uttrycka sina tankar utanför ramen av de förbestämda frågorna och teman om det är intressant och/eller relevant för studien. Intervjun inleds i enlighet med den etik som har tagits upp under punkt 3.6. Syftet med studien kommer att tas upp, hur resultaten kommer att användas och att respondenten deltar i studien frivilligt.

Intervjuerna blev alltså semistrukturerade och de användes för att samla in data, där respondenterna utöver de förberedda utgångsfrågorna i intervjuguiden även hade möjlighet att fritt utveckla sina svar. Denna flexibilitet syftade till att höja kvaliteten på den insamlade datan och att stimulera till diskussion, vilket potentiellt kunde belysa nya teman och perspektiv som inte förutsågs vid utformningen av intervjuguiden. Även om inte alla frågor ställdes ordagrant, säkerställdes att alla planerade teman behandlades och att alla respondenter

adresserade frågeställningen. Denna ansats underlättade en djupare förståelse och berikade dataanalysen genom att inkludera oväntade insikter och perspektiv.

Tabell 3: Intervjuguide

Tema	Exempelfrågor
Etik	● Skulle du vilja vara anonym i denna intervju? ● Är du frivilligt med i studien? ● Godkänner du att intervjun spelas in?
Bakgrund och Erfarenhet	● Kan du berätta lite om din bakgrund inom programmering? ● När började du använda dig av ChatGPT? ● Vad är en bra prompt enligt dig?
ChatGPT som ett verktyg inom Systemutveckling & Utmaningar	● Hur ofta använder du ChatGPT inom programmering? ● Hur använder du ChatGPT inom programmering? ● Brukar du ofta få det svar du önskar från ChatGPT? ● Upplever du att ChatGPT hallucinerar ibland? ○ Vad tror du det beror på isåfall? ● Hur hjälper ChatGPT dig att lösa programmeringsproblem? ● Vilken LLM använder du mest när du programmerar?

Prompt Engineering	• Vad betyder Prompt Engineering enligt dig? • Är du bekant med några prompt tekniker? (Till exempel CoT eller few-shot) • Använder du dig av prompt-tekniker? • Kan du berätta hur din prompt brukar se ut när du använder ChatGPT för att lösa ett programmeringsproblem? • Känner du någon skillnad på svar från ChatGPT beroende på hur du promptar? • Tror du prompt tekniker kan vara hjälpsamma i större, mer komplexa problem? • Tror du prompt engineering är viktigt för att få ett bra svar från LLMs?

3.2.4 Intervjuns kontext

Jacobsen (2017) beskriver två olika typer, även kallat kontext, av platser en intervju kan genomföras. Dessa kontexter är antingen “naturlig” eller “artificiell”. Den naturliga kontexten innebär att personen, som Jacobsen skriver, ska “känna sig hemma”. Det är intervjun och dess innehåll som avgör var den kontexten ska utföras. T.ex. är en intervju som undersöker arbetsförhållanden lämplig att vara ute på en arbetsplats. Därav blir den naturliga kontexten arbetsplatsen. Den artificiella kontexten är en plats som varken undersökaren eller respondenten har någon direkt relation till, exempelvis undersökarens arbetsplats. Jacobsen (2017) förklarar att innehållet i en intervju kan påverkas av den kontext som den utförs i, något som Jacobsen kallar “kontexteffekter”. Vidare beskrivs olika problem som kan stötas på för respektive kontext. Den artificiella kontexten kan resultera i att tillgjorda svar ges av respondenten och den naturliga kontexten, om den utförs i exempelvis ett hem, störas av till exempel barn, telefonsamtal etc.

Utifrån detta har kontexten för intervjun valts till att vara naturlig. Eftersom det är studenter som intervjuas valdes den naturliga kontexten till att vara på Ekonomihögskolan i Lund, en plats där undervisning för studenterna bedrivs vilket resulterar i att de ´känner sig hemma´ i den miljön. Detta kommer säkerställa att de svar respondenterna ger inte påverkas på ett dåligt sätt av den valda kontexten. I och med detta kommer intervjuerna att ske

´ansikte-mot-ansikte´. Den typ av intervju beskrivs enligt Jacobsen (2017) som ett bra tillvägagångssätt för att ge kommunikationen mellan undersökare och respondent ett bra utgångsläge för utbyte av information, vilket det inte går att få i samma grad jämfört med exempelvis mejlintervju eller över telefon.

3.2.5 Ljudinspelning

Jacobsen (2017) beskriver att ögonkontakt är viktigt för att utföra en bra intervju. Ett problem som kan uppstå då är att samtidigt kunna utföra anteckningar under tiden. Jacobsen (2017) förklarar att detta på ett enkelt sätt kan fixas genom att spela in intervjuerna. På så sätt får man också med allt respondenten säger vilket är möjliggör för en utförlig transkribering.

I enlighet med vad Jacobsen (2017) beskriver angående ljudinspelning har samtliga intervjuer valts att spela in efter samtycke från respondenterna. Detta underlättar för att göra utförliga transkriberingar, inte missa viktiga detaljer under intervjun samt att hålla ögonkontakt med respondenten. Samtliga intervjuer visas enligt tabellen nedan med information om vilken respondent det är, vilken kontext intervjun har samt vilken tid varje ljudinspelning från intervjun har.

Tabell 4: Ljudinspelning

Respondent	Kontext	Längd
R1	Ansikte–mot-ansikte	19 minuter 7 sekunder
R2	Ansikte–mot-ansikte	19 minuter 47 sekunder
R3	Ansikte–mot-ansikte	23 minuter 36 sekunder
R4	Ansikte–mot-ansikte	17 minuter 47 sekunder

3.3 Analysprocessen

3.3.1 Förberedelse för analys

Tidigare i metodavsnittet har fördelarna med kvalitativ metod, såsom att datan insamlad kan vara djupare och komplexare än bara statistik. Däremot kan bearbetning av denna data vara bekymmersam. För varje timme av inspelad intervju tar det 4 till 5 timmar att transkribera (Oates et al. 2022) vilket gör det tidskrävande att få ner all data. Efter den tidskrävande processen är det reducera datan då Oates et al. (2022) menar på att den råa datan från intervjuer kan delas upp i tre kategorier: segment av intervjun som inte kommer att behövas alls till studien, segment av deskriptiv information till kontexten av forskningen och segment av värdefull data som kommer att användas till slutsatsen. Vidare beskriver författarna att mycket av analysen ligger på forskarna själva. Det är forskarna som måste se mönster och sammanhängande data.

Jacobsen (2017) beskriver att transkribering av en ljudinspelning till text är fördelaktigt när det kommer till att gå igenom materialet från själva intervjun. Om en transkribering inte genomförs kan det ta lång tid att behöva lyssna igenom en ljudinspelning och dessutom om det behövs göras flera gånger.

I enlighet med Jacobsen (2017) transkriberades alla genomförda intervjuer. Transkriberingarna gjordes med hjälp av Whisper som tillhandahålls av Open AI, men också en manuell genomgång för att säkerställa att transkriberingen gjord av Whisper är korrekt. Tabellen nedan visar vilken transkriberad text från intervjuerna som hör ihop med respektive respondent. Dessa transkriberingar finns tillgängliga som Appendix A-D.

Tabell 5: Transkriberingar

Respondent	Transkribenter	Appendix
R1	Jesper Hultkrantz & Jonatan Carlsson	A
R2	Jesper Hultkrantz & Jonatan Carlsson	B
R3	Jesper Hultkrantz & Jonatan Carlsson	C
R4	Jesper Hultkrantz & Jonatan Carlsson	D

3.3.2 Genomförande av analys

Jacobsen (2017) beskriver att när en analys, av det innehåll som samlats in från intervjuer, ska genomföras kan en uppdelning av innehållet göras. Innehållet delas upp i mindre kategorier för att kunna samla liknande data under en specifik kategori. Jacobsen (2017) kallar detta för en "öppen kodning". En sådan kodning förenklar analysen eftersom det är tydligt vilken data som berör vilken kategori. De huvudsakliga kategorierna beskrivs enligt Jacobsen (2017) att bestå av de teman som återfinns i intervjuguiden. Dessa kategorier är generella och för att få mer konkreta kategorier delar man in dessa i vad som kallas "underkategorier". Dessa kategorier kan sedan presenteras enligt en definitionstabell vilket underlättar för de som utför undersökningen att få en överblick av vilken data som ska tillhöra vilken kategori.

I enlighet med vad Jacobsen (2017) beskriver angående analysering av data har ett antal kategorier samt underkategorier tagits fram. Kategorierna grundar sig utifrån de teman som återfinns i intervjuguiden och underkategorierna har tagits för att få mer konkreta områden att fokusera på. Dessa kategorier samt underkategorier presenteras enligt definitionstabellen nedan. Varje kategori har också blivit tilldelad en specifik färg. Färgerna används för att markera data, i de transkriberade intervjuerna, som tillhör en specifik kategori. För att hitta data från intervjuerna som tillhör en specifik kategori lästes varje transkribering igenom noggrant. Efter detta erhölls en tydlig bild av vad respondenten sagt och datan tilldelas en specifik färg. Tilldelningen skedde utifrån vilken fråga som ställdes samt genom tolkning av svaren. Detta underlättade analysen eftersom data om en specifik kategori är samlat under en specifik färg.

Sammanställningen av datan, se punkt 4, presenteras enligt de kategorier samt underkategorier som tagits fram. Varje respondents svar som berör en kategori presenteras i den ordning som visas enligt definitionstabellen nedan. Detta för att resultatet ska vara i enlighet med tabellen, vilket i sin tur gör det lättare för läsaren att hänga med och förstå upplägget av det presenterade resultatet.

Tabell 6: Definitionstabell

Kategorier	Underkategorier	Färg
Bakgrund och erfarenhet	• Bakgrund inom programmering	Grön
ChatGPT inom systemutveckling	• Användningsområden av ChatGPT inom programmering • Begränsningar med användandet av ChatGPT inom programmering • Tillvägagångssätt för användning av ChatGPT inom programmering	Blå
Prompt engineering	• Kännedom om prompt engineering och prompt tekniker • Användning av prompt tekniker	Gul

3.4 Generaliserbarhet

Att använda kvalitativa datainsamlingsmetoder i denna undersökning gör frågan om generaliserbarhet särskilt relevant. Generaliserbarhet, ett nyckelbegrepp inom forskning, berör frågan om i vilken mån resultaten från en given studie kan tänkas vara tillämpbara utanför den omedelbart undersökta kontexten eller gruppen av deltagare. Det innebär att överväga möjligheten att applicera studiens slutsatser på en bredare population eller i olika scenarier och miljöer än de som direkt observerades (Lee & Baskerville, 2003). Lee och Baskerville (2003) utforskar detta koncept och understryker dess betydelse, särskilt i ljuset av kvalitativ forsknings natur där datainsamlingen fokuserar på djupgående förståelse av specifika fenomen inom deras naturliga sammanhang.

I artikeln betonas vidare vikten av att kritiskt granska och tydliggöra begreppet generaliserbarhet genom att presentera en ram för att klassificera dess olika former. Denna ram organiserar generaliserbarhet i fyra typer, vilket gör det möjligt för forskare att på ett mer precist sätt identifiera vilken form av generaliserbarhet som är lämplig för deras studie. Denna distinktion bidrar till en djupare förståelse för hur forskningsresultat kan ha relevans utöver den omedelbart undersökta kontexten, vilket är särskilt värdefullt i kvalitativ forskning där specifika situationer och erfarenheter utforskas i detalj. Genom att klargöra olika aspekter av generaliserbarhet, erbjuder Lee och Baskerville (2003) ett verktyg för forskare att kritiskt

utvärdera och argumentera för sin studiens relevans och tillämpbarhet på bredare sammanhang.

Denna förståelse för generaliserbarhetens komplexitet, som framhävs av Lee och Baskerville (2003), stärker argumentet för den kvalitativa studiens potentiella bidrag till kunskapsfältet. Det understryker att även om en studie fokuserar på specifika fall eller erfarenheter, kan dess insikter och teorier ha ett värde som sträcker sig bortom dessa omedelbara gränser, genom att belysa universella teman eller djupare mekanismer som kan vara relevanta i liknande men skilda kontexter.

3.5 Validitet och reliabilitet

3.5.1 Validitet

Jacobsen (2017) beskriver två krav när empiri ska samlas in för en undersökning. Giltighet och relevans är ett av dessa krav och är synonymt med ordet validitet. Validitet syftar till, utifrån de frågor undersökningen ska besvara, att den insamlade empirin ger svar på det som efterfrågas. Jacobsen (2017) menar på att det finns olika typer av giltighet och relevans (validitet), närmare bestämt två stycken. Dessa typer är intern giltighet och extern giltighet. Intern giltighet syftar till verkligheten och hur den beskrivs av en forskare. Är forskaren resultat riktigt och uppfattningen av dem det också. Extern giltighet syftar till generalisering av resultaten. Detta avser hur väl en generalisering kan göras av en undersöknings resultat.

I den här uppsatsens studie förbättras den interna giltigheten genom att fyra studenter från samma program har intervjuats. De har erfarenheter som är likvärdiga vilket ger en förståelse för vilket sammanhang de är tagna ifrån. Den externa giltigheten stärks inte i den här studien. Detta eftersom respondenterna är tagna från en grupp som kanske inte är representativ utanför deras klass och utbildningen.

3.5.2 Reliabilitet

Jacobsen och Hellström (2002) betonar att för att en studie ska anses pålitlig, är det essentiellt att dess empiriska grund är både tillförlitlig och trovärdig. Detta innebär att forskningen måste utföras på ett sätt som skapar förtroende för studiens resultats äkthet. En viktig aspekt är avsaknaden av signifikanta mätfel; närvaron av sådana fel undergräver studiens reliabilitet. Enligt Jacobsen och Hellström (2002) indikerar en korrekt utförd undersökning en möjlighet till reproducerbarhet, vilket innebär att om samma studie genomfördes återigen under identiska förhållanden, bör resultaten kunna replikeras. Med metodvalet kvalitativa intervjuer kan det vara svårt att upprätthålla reliabilitet över tid då respondenterna kan ändra sina åsikter och svar eller minnas sitt svar som de gav första intervjun och svara samma sak (Oates et al. 2022).

I valet av metod, som består av en kvalitativ intervjustudie, säkerställs att de empiriska data som samlas in är tillförlitliga och trovärdiga, vilket bidrar till en högre grad av studiens reliabilitet. Potentiella svagheter i studien kan knytas till den kvalitativa intervjun, där resultaten över tid kan vara utsatta för förändringar.

3.6 Etik

Jacobsen (2017) beskriver tre olika etiska krav som är essentiella när det kommer till förhållandet mellan intervjuaren och respondenten. Dessa tre krav är ”informerat samtycke”, ”rätten till privatliv” och ”krav på korrekt presentation av data”.

Informerat samtycke

Det första kravet, informerat samtycke, handlar om att den som deltar i en undersökning frivilligt ska förstå vad sitt deltagande innebär. Det berör aspekter såsom att den som undersöks inte ska känna sig tvingad till att vara med, att få information om syftet och användningen av undersökning respektive resultaten samt att den som undersöks också förstår den information den har tagit del av Jacobsen (2017). Detta krav har följts genom att presentera information om syftet och hur resultaten ska användas till de som ska undersökas. De får också information om att delta i undersökningen är frivilligt och att det är tillåtet att avbryta intervjun om så önskas.

Rätten till privatliv

Det andra kravet, rätten till privatliv, handlar om att den som undersöks inte ska behöva ge ut information om sådant som anses vara av privat natur. Det berör aspekter såsom att frågor som ställs till den som undersökts kan uppfattas som känsliga och att den som undersökts kan identifieras utifrån det material som presenteras i undersökningen Jacobsen (2017). Detta krav har följts genom att de undersökta blev informerade om att deras deltagande kommer att vara anonymt och att det inte kommer att gå att identifiera dem utifrån materialet i uppsatsen. Detta kommer att säkerställas genom att inte utge några namn eller andra unika personliga identifierare.

Krav på korrekt presentation av data

Det tredje kravet, krav på att man blir korrekt återgiven, handlar om att de resultat som presenteras i uppsatsen ska vara korrekta. De ska inte ändras eller manipuleras på något sätt så att resultaten visar något annat än vad de faktiskt säger (Jacobsen, 2017). För att kunna följa det här kravet och att de resultat som presenteras ska vara korrekta kommer ett noggrant arbete göras för att förhindra felaktig återgivning av resultaten.

3.7 AI

ChatGPT 3.5 och 4 har använts som ett hjälpmedel för att skriva den här uppsatsen. AI har använts för att förbättra språk och komma på idéer utifrån den text som vi själva skrivit. Den här uppsatsen är i klar majoritet mänskligt skapad och ChatGPT har mer använts som ett hjälpmedel för att förbättra den text som vi redan skrivit. Vid utformandet av titeln och forskningsfrågan har ChatGPT använts för att ge förslag på olika titlar och forskningsfrågor tills vi fick fram ett svar som ansågs bäst passa in i uppsatsen. Hjälp från ChatGPT med meningsuppbyggnad och förtydligande av redan skriven text har använts till viss del genom hela uppsatsen. För att automatisera transkriberingen har vi använt oss av Whisper, en mjukvara som transkriberar med hjälp av AI.

4 Empiriskt resultat

4.1 Bakgrund och erfarenhet

4.1.1 Bakgrund inom programmering

Det respondenterna beskriver visar att de har olika bakgrunder inom programmering. Respondent R1 har en bakgrund inom programmering som sträcker sig tre år tillbaka. R1 har hållit på med objektorienterad programmering i Java och C# och fått sin erfarenhet från utbildningen på det systemvetenskapliga kandidatprogrammet. Respondenten har även läst en sommarkurs i Python samt har jobbat som labbhandledare.

> *"Ja, jag har programmerat i tre år, objektorienterat i Java och C Sharp och fick min grund här på institutionen för informatik på det systemvetenskapliga kandidatprogrammet. Jag hade ingen erfarenhet av programmering innan det. Sen har jag också tagit lite sommarkurser i Python, men det är min erfarenhet. Sen är jag också labbhandledare på institutionen. Så jag har fått lite mer erfarenhet av programmering det senaste året genom att hjälpa studenter på labbar"* (Appendix A, #3).

Respondent R1 håller också på med programmering på sin fritid.

> *"Jag gillar att programmera på fritiden också och är genuint intresserad av det. Följer lite trådar och så på Twitter om programmering."* (Appendix A, #7).

Respondent R2 började först programmera i samband med starten av det systemvetenskapliga kandidatprogrammet. R2 sitter på fritiden och programmerar likt R1.

> *"Programmering, det har jag inte hållit på med innan….Så för programmering så har jag framförallt jobbat här på institutionen då, med Java, C# , lite SQL, och det är det som kommer med utbildningen egentligen. Ja, och sen har jag blivit lite bättre inom det också genom att jobba som lab-assistent. Så jag har lärt mig mycket där också."* (Appendix B, #5).

R2 sitter på fritiden och programmerar likt R1.

> *"Ja, men ibland gör jag det. Bygger lite små applikationer. Jag satt och gjorde någon app i, vad heter det? En typ Calc eller en budgetapp typ, för skojs skull. Så det är kul tycker jag."* (Appendix B, #11).

Respondent R3 har också fått sin bakgrund från systemvetenskapliga kandidatprogrammet likt R1 och R2.

> *"Ja, min bakgrund inom programmering är då från min utbildning på systemvetenskapliga kandidatprogrammet. Jag har inte programmerat innan dess, så det är den erfarenheten jag har av programmering."* (Appendix C, #4).

R3 nämner också att den har erfarenhet inom programmeringsspråken C#, Java. Likt R1 har också R3 erfarenhet inom Python, men också Javascript.

> *"Ja, det är C# , det är Java. Jag har också läst en kurs på Python, så jag kan lite Python där också. Och även lite scripting-språk som Javascript till exempel."* (Appendix C, #6).

R3 sitter inte så mycket på fritiden och programmerar.

> *"Nej, inte direkt så mycket på fritiden. Väldigt lite i så fall."* (Appendix C, #10).

R4, likt R1, R2 och R3, har också fått sin erfarenhet inom programmering i samband med sina studier inom det systemvetenskapliga kandidatprogrammet. R4 nämner också att viss programmering på fritiden förekommer.

> *"Ja, mest i skolan ska jag säga. Men lite har jag hållit på på fritiden."* (Appendix D, #6).

Sammanfattningsvis har samtliga respondenter liknande erfarenheter inom programmering där de började programmera i samband med sina studier på det systemvetenskapliga kandidatprogrammet. R1 och R2 har ytterligare praktisk erfarenhet från att ha jobbat som labbhandledare, vilket R3 och R4 inte har gjort. R1, R2 samt R4 säger att de sitter på fritiden och programmerar medan R3 inte gör det.

4.1.2 Bakgrund inom användning av ChatGPT

Respondenternas erfarenhet när det kommer till att använda sig utav ChatGPT varierar, men alla har på något sätt använt och tagit del av verktyget inom sin egen programmering. R1 började använda ChatGPT direkt efter att det lanserades i december 2022 och har använt det dagligen efter det.

> *"ChatGPT började jag använda direkt när den lanserades i december 2022 ungefär och har använt det dagligen sedan dess."* (Appendix A, #11).

R1 menar på att sitt eget användande av ChatGPT har ökat efter att institutionen uppmuntrade till att använda det.

> *"Efter att studierektorn och lärarna på institutionen började uppmana oss att använda AI-modellen så har jag använt det mycket mer."* (Appendix A, #29)

Respondent R2 blev introducerad till ChatGPT av en vän och började använda det för programmering i mars 2023,

> *"Jag var lite sen på det, men ändå ganska nära anslutning till att det kom ut. Jag minns att min polare visade typ ett julrim som han hade skrivit med mig på ChatGPT och jag bara, åh shit, den kan generera mycket text den här. Så det var så jag kom i kontakt med det först. Sen började jag använda det för programmering typ mars 2023."* (Appendix B, #13).

R3 började använda ChatGPT ganska nära inpå efter att det lanserades och har använt det kontinuerligt efter det. R3 säger att för programmering började ChatGPT användas under en kurs på vårtermin under utbildningen.

> *"Jag började använda ChatGPT för programmering när jag började med programmeringskurserna på vårterminen."* (Appendix C, #20).

R4 säger inte uttryckligen när sitt användande av ChatGPT för programmering började men använder det nästan hela tiden när R4 programmerar för att förstå och lösa problem. R4 säger:

> *"Nästan hela tiden. För att få backa upp och förstå allting jag gör. Eftersom jag inte har gjort det tidigare. Så jag vill alltid förstå vad jag gör."* (Appendix D, #22).

Sammanfattningsvis använder alla respondenter sig av ChatGPT när de programmerar. R1, R2 och R3 började använda sig av ChatGPT nära efter lanseringen. R2 och R3 började uttryckligen använda ChatGPT för programmering under vårterminen 2023.

4.2 ChatGPT inom programmering

4.2.1 Användningsområden av ChatGPT inom programmering

Respondenterna använder ChatGPT inom flera olika områden inom programmering. De främsta användningsområdena är kodgenerering, felsökning och inlärning av nya koncept inom programmering. R1 uttrycker att ChatGPT är bäst när det kommer till att automatisera monotona steg inom programmering men också för felsökning.

> *"Jag tycker att ChatGPT är bäst på att automatisera de monotona stegen i utvecklingen. Typ att generera en basklass"* (Appendix A, #17).

> *"Felsökning är den riktigt bra på."* (Appendix A, #47).

R2 beskriver att ChatGPT, inom programmering, används främst för att generera "boilerplate-kod".

> *"Då använde jag det mest för att generera lite kod. Alltså ganska. Vad kan man säga? Boilerplate-kod."* (Appendix B, #15).

R2 beskriver också att det används för felsökning.

> *"Och framförallt för felsökning"* (Appendix B, #15).

R2 uttrycker också att ChatGPT är bra på att lära sig nya saker inom programmering som man inte har så bra koll på.

> *“Men också upplärning. Alltså, som sagt, den är jättebra på att förklara kod.”* (Appendix B, #41).

R3 säger att ChatGPT kan användas för att få förslag på hur en metod kan se ut samt få en förklaring av den.

> *"Just för programmering då så handlar det mycket om att få förslag på hur metoder kan se ut och få metoder förklarade."* (Appendix C, #24).

R3 beskriver också att ChatGPT är bra på att automatisera kod.

> *"Säg att du ska göra 10 metoder i en klass och de här metoderna ska se i princip likadana ut i fem andra klasser. Att kunna automatisera, det är den bra på, jag slipper skriva det själv."* (Appendix C, #38).

R3 beskriver ytterligare ett användningsområde för ChatGPT vilket är felsökning.

> *"Felsökningar, varierande. Det handlar också tror jag om kunskap om problemet från början då. Vad det är för fel som uppstår när man kör programmet till exempel."* (Appendix C, #38).

R4 beskriver att ChatGPT används mycket för att generera kod så man slipper göra det själv.

> *"Mycket så här om man ska lägga in massa exempelobjekt till en databas så vill jag generera massa för att det är skönt att slippa göra själv,"* (Appendix D, #24).

Debugging var något R4 gjorde i början med hjälp av ChatGPT men inte så mycket längre.

> *"Debugging i början. Nu vill jag säga att jag är bättre på det att göra själv. Det är ofta lättare i mjukvaran du arbetar i att hitta problemen än GPT."* (Appendix D, #24).

R4 beskriver ytterligare att ChatGPT används pedagogiskt verktyg som fungerar lite som en privatlärare

> *"Jag tycker att det är ett väldigt bra verktyg. Jag tycker att det är pedagogiskt. Och som en liten privatlärare. Det är ju inte alltid helt rätt. Men man får ta det med en nypa salt. Och jag tycker att det har hjälpt mig att utvecklas i programmering. För det förklarar väldigt tydligt när man ställer frågan vad det är som händer i koden."* (Appendix D, #16).

Sammanfattningsvis uttrycker samtliga respondenter att de använder ChatGPT främst för två saker, att generera monoton kod samt för felsökning. R3 beskriver också att ChatGPT kan användas för att få förslag på hur en metod kan se ut, lite som brainstorming. R4, likt R2, uttrycker att ChatGPT är ett bra pedagogiskt verktyg som fungerar lite som en privatlärare som gör att man kan lära sig nya saker.

4.2.2 Begränsningar med användandet av ChatGPT inom programmering

Respondenterna har identifierat ett antal olika begränsningar med användningen av ChatGPT inom programmering. Dessa begränsningar är främst hallucinationer men också egen förståelse för ett problem samt att det kan hindra ens eget lärande.

R1 nämner att det varit problem med att ChatGPT inte följer det som står i prompten och hallucinerar.

> *"Ja, om man bett den göra ett litet kodstycke i Python där man har specificerat CSV, så har den gett ut ett exempel på Pandas.Vilket är inte helt fel men det är bara ett annat tillvägagångssätt men det var inte det man var ute efter"* (Appendix A, #41).

R2 beskriver likt R1 att ChatGPT kan hallucinera.

> *"Problemet i början där var ju att ChatGPT kanske inte var så optimerad. Den var ju bra, men det var inte helt 100 optimerad och jag var inte jättebra på att prompta. Så det kunde lätt bli att jag gick i cirklar med problem. Alltså flera gånger som jag skickade in kod, skrev lite halvdålig input och så fick jag tillbaka näst intill samma kod. Och så skrev jag nej, det här är fel, fick jag tillbaka samma. Så det blev liksom AI-hallucinationer på att det här är rätt när det inte var rätt."* (Appendix B, #15).

R3 säger också att ChatGPT kan hallucinera i vissa fall.

> *"Det har hänt att den har outputat, liksom den har vetat om vilket bibliotek jag använder. Men den har outputat en klass som inte finns, som inte har den funktionaliteten överhuvudtaget. Och verkligen satt sig vid det, att okej, det här finns, gör så här. Så det är det främst för hallucinationer, som jag har märkt."* (Appendix C, #44).

R3 beskriver ytterligare att om man har dålig förkunskap inom ett programmeringsspråk så kan man fastna när man använder ChatGPT.

> *"Men om jag har dålig förkunskap inom programmeringsspråket eller ramverket så kan det vara ett hinder. Man fastnar i det för länge. Man liksom, ja, ChatGPT ska lösa det åt mig. Då blir det att man hamnar i en liten ond cirkel nästan."* (Appendix C, #48).

R4 beskriver att ChatGPT ger fel svar 1 av 10 gånger men också att det kan hämma ens eget lärande om man låter ChatGPT göra allting åt en.

> *"om jag inte skulle ha självkontroll och bara använda den och tro att den gör allting rätt så skulle den ju hämmar mig på något sätt i att jag inte lär mig."* (Appendix D, #44).

Sammanfattningsvis upplever samtliga respondenter att ChatGPT kan hallucinera. R3 beskriver också att förkunskapen inom ett programmeringsspråk kan vara ett hinder när man använder ChatGPT. R4 beskriver också att ChatGPT kan hämma ens eget lärande om man låter den göra allting och tror att allt är rätt.

4.2.3 Tillvägagångssätt för användning av ChatGPT inom programmering

Respondenternas tillvägagångssätt när de använder ChatGPT visar ett flertal olika strategier. R1 beskriver att för felhantering kopieras felkoden och ett kort meddelande skrivs.

> *"Oftast i början så kan det hända att man bara skickar in så lite som man behöver för man vill alltid att det ska gå snabbt. Då kan det hända att man bara skickar in felkoden och skriver ett kort meddelande, jag fick detta felmeddelande vad ska jag göra."* (Appendix A, #49).

R1 fortsätter beskriva att om ChatGPT inte ger rätt svar så utformas en mer utförlig prompt.

> *"Men om man inte får rätt svar första gången så brukar ju kvaliteten till prompten öka och man kanske beskriver vad det är man gör och man kanske skickar in koden från andra klasser som berör problemet."* (Appendix A, #49).

R2 använder ChatGPT som en mentor för att lösa exempelvis ett problem och få en bättre förståelse för vad problemet är och hur man kan lösa det.

> *"att man kan använda det som en mentor. För att okej, hur kan jag lösa det här problemet? Och så kanske man får några steg som man kan använda för att lösa problemet."* (Appendix B, #19).

> *"ifall man har ett kodstycke i återigen ett språk man inte kan. Så kan man skicka in det och säga vad gör det här kodstycket? Så förklarar den steg för steg vad det här kodstycket gör."* (Appendix B, #23).

R3 beskriver att när förståelsen för ett problem är väldigt litet så litar man på svaret från ChatGPT helt och en kort prompt som bara säger att den ska lösa problemet används.

> *"om jag inte har en aning då kan det bli att man litar på ChatGPT's output helt egentligen. Att man frågar, lös det här åt mig så får man ut någonting. Och så tar man i princip det."* (Appendix C, #26).

Ytterligare beskriver R3 ett tillvägagångssätt för att hantera om ChatGPT ger dig helt fel svar. Det handlar om att få ChatGPT att komma ifrån den lösning som den tror är rätt för problemet.

> *"då blir det ofta så här att jag säger till den bara okej, alltså den hittar inte den här klassen. Finns det någon annan klass, något annat bibliotek jag kan använda mig av? Så att den kommer ur det här spåret att du ska använda det här biblioteket och den här klassen. Så kanske den hittar ett annat bibliotek som faktiskt har funktionalitet för det här, liksom har den rätta klassen."* (Appendix C, #46).

R3 beskriver också att kontexten är viktig när ett programmeringsproblem ska lösas.

> *"Man kan specificera till exempel programmeringsspråk och specificera också hur resten av, säg att du har flera klasser, specificera hur resten av strukturen på projektet ser ut. Också specificera, kanske om du har kursmaterial där de vill att du ska lösa på ett visst sätt, specificera det så att den vet hur du vill lösa problemet så att den använder sig av de biblioteken som önskas."* (Appendix C, #62).

R4, likt R1, vid felsökning kopierar in felmeddelandet till ChatGPT för att få förslag på vad felet kan vara och hur man löser det.

> *"Om jag inte lyckas lösa det där så tror jag att jag brukar nästan bara copy pastea fel kod, kolla mycket GPT, den brukar ge massa olika förslag."* (Appendix D, #50).

R4 beskriver också att om svaret ChatGPT inte ger är tillräckligt bra så ges tydliga instruktioner för hur outputen ska se ut.

> *"om jag är inte nöjd med svaret jag får så bryter jag ner det och ger den tydliga instruktioner i hur den ska ge mig output."* (Appendix D, #80).

Sammanfattningsvis har samtliga respondenter ett liknande tillvägagångssätt när de försöker lösa något de inte förstår med hjälp av ChatGPT. De skriver alla en kort prompt, såsom "lös det här åt mig" eller "jag fick detta felmeddelande vad ska jag göra". R1 beskriver att om det första svaret från ChatGPT inte är bra så skrivs en tydligare prompt som förklarar problemet mer detaljerat. R3 beskriver att man försöker få ChatGPT att gå ifrån det felaktiga svaret genom att i sin prompt specifikt fråga efter en annan lösning.

4.3 Prompt Engineering

4.3.1 Kännedom om prompt engineering och prompt tekniker

Respondenterna har olika kännedom om prompt engineering och prompt tekniker. Deras kännedom varierar från lite kännedom till lite mer kännedom. R1 beskriver sin kännedom om prompt engineering som hur man strukturerar prompts. R1 ger följande definition:

> *"För mig betyder prompt engineering hur man strukturerar och bygger prompts.För att få bästa möjliga svar. Hur man skriver bra prompts för att få bra svar."* (Appendix A, #53).

Vid frågan om R1 känner till någon speciell prompt teknik gavs detta svar:

> *"Nej, det skulle jag inte säga. Jag kanske är det undermedvetet men inte som jag kan nämna namnet på"* (Appendix A, #59).

R2 ger ingen direkt definition om promp engineering men ger en förklaring om vad en bra prompt är men liknar den R1 ger till viss del.

> *"En prompt är ju alltså det man skickar in, alltså inputen till AI:n för att få outputen."* (Appendix B, #51).

R2 beskriver sig vara bekant med en prompt teknik som heter Chain of Thought men vet inte direkt vad den innebär.

> *"Jag känner igen Chain of Thought men jag är inte så bekant med det att jag vet vad det är."* (Appendix B, #59).

R3, likt R1, ger en definition om vad prompt engineering är.

> *"alltså hur man strukturerar sina frågor för att få ut det svaret man vill ha."* (Appendix C, #56).

Vid frågan om R3 känner till någon prompt teknik, såsom Chain of Thought eller Few-Shot, svarade respondenten att ingen sådan förståelse fanns.

> *"Nej, inte direkt."* (Appendix C, #68).

R4 ger också definition om prompt engineering som liknar R1 och R3.

> *"hur du promptar på ett bra sätt för att AI:n ska förstå dig och ger dig ett resultat du vill ha väldigt snabbt."* (Appendix D, #20).

R4 är inte heller, likt R1 och R3, bekant med någon specifik prompt teknik.

> *"Nej, egentligen inte."* (Appendix D, #70).

Sammanfattningsvis har samtliga respondenter en liknande kännedom om vad prompt engineering är. De ger liknande definitioner för begreppet och förklarar det som att det är hur man bygger upp en prompt för att få det svar man önskar från ChatGPT. Respondent R1, R3 och R4 har ingen tidigare kännedom om någon prompt teknik. R2 beskriver sig känna igen tekniken Chain of Thought men har ingen förståelse för vad tekniken innebär.

4.3.2 Användning av prompt tekniker

Utifrån vad respondenterna sagt förekommer viss användning av prompt tekniker. Teknikerna Chain of Thought och Few-Shot har indirekt använts flera fall.

Efter att respondent R1 fick ta del av en kort beskrivning om prompt teknikerna Few-Shot samt Chain of Thought visade det sig att R1 hade använt sig av de teknikerna inom programmering. R1 gav följande beskrivning:

> *"Ja, exempel har jag använt speciellt i programmering, men också när jag skriver texter. Där tror jag mer än i programmering så brukar jag oftast lägga in en exempeltext och beskriva att det är den här strukturen jag vill ha på texten och sen så ger den information som berättar vad den ska göra."* (Appendix A, #61).

> *"Det skulle jag säga att jag har, mer i programmering då. Speciellt om jag har ett större problem så har jag väl försökt dela upp det för att öka sannolikheten för att jag får rätt svar. Det är väl tanken bakom det, att man har ett mindre problem så att man kan fokusera på det."* (Appendix A, #65).

R2 beskrev att det kan ha förekommit användning av prompt tekniken Chain of Thought efter att R2 tagit del av en kort beskrivning om de två teknikerna.

> *"Det kan jag nog ha gjort ja. Det är nånting som man stöter på i felsökning."* (Appendix B, #61).

Efter frågan om R2 använder sig utav prompt tekniken Few-Shot svarade respondenten att det kan ha förekommit sådan användning omedvetet.

> *"Eller kanske. Att återigen omedvetet. Fast då har jag inte gett mer. Jag tror inte jag har gett mer än ett exempel när det har varit ett problem."* (Appendix B, #65).

R3 nämner att användning av Chain of Thought har förekommit men att det inte är något som R3 aktivt tänker på.

> *"Jo, men det har jag väl. Jag kan inte liksom på rak hand säga något specifikt exempel, men det blev lite av det man gör när man har ett större projekt. Mest är det*

> *inget jag aktivt tänker på. Jag har inte tänkt på den här tekniken aktivt."* (Appendix C, #70).

Efter att respondenten reflekterat över tekniken beskriver R3 att en sådan teknik kan göra det lättare för ChatGPT att lösa ett problem och göra lättare att förstå själv också..

> *"Ja, men det tycker jag att överlag känns som att det blir lättare för ChatGPT att lösa mindre delar av ett problem i och med att den har bara så mycket kontext."* (Appendix C, #72).

> "*om jag ber den förklara varje del så blir det ju att man bryter ner det för sig själv också i huvudet.*" (Appendix C, #74).

Vid användning av Few-Shot tekniken beskriver R3 att sådan användning kan ha förekommit.

> "*Det kan det också vara att man kastar in exempel, till exempel då kursmaterial? I så fall så ja.*" (Appendix C, #76).

R3 har inte uttryckligen använt sig av Few-Shot tekniken men följande beskrivning av respondenten är en indirekt användning av tekniken.

> *"Säg att det är en metod jag vill ha som ska göra en viss sak. Då säger jag att jag vill ha en metod i det här språket. Den ska heta det här. Den ska returnera det här. Och den ska göra det här."* (Appendix C, #80).

Vid frågan om R4 har använt sig av Chain of Thought tekniken svarade respondenten att det kan ha förekommit sådan användning indirekt.

> "*Ja, intuitivt skulle jag säga att jag har gjort det. Inte vet jag metoden, men det har jag väl gjort. Om det är ett för stort problem så är det lättare att den förstår lättare om jag bryter ner det, absolut.*" (Appendix D, #72).

Vidare förklarar R4 att använda sig av en sådan teknik ökar den egna förståelsen.

> *"om den inte ger mig en bra förklaring med det större problemet och jag bryter ner det i mindre delar så att jag förstår det bättre också, absolut. Då förstår jag ju bättre."* (Appendix D, #74).

R4 använder sig inte av Few-Shot tekniken i någon större utsträckning men ibland förekommer det att R4 i sin prompt skriver att ChatGPT ska ge en viss kod samt en förklaring till den.

> *"Ja, det kan jag väl göra ibland. I början när jag inte visste så mycket mer och jag ville ha en hel kod. Till en klass till exempel, så ville jag att den skulle förklara. Ge mig den här koden och beskriv varenda steg du genererar. Men mer avancerat än så har jag inte gett den instruktioner i output."* (Appendix D, #78).

Sammanfattningsvis använder samtliga respondenter i någon utsträckning sig av de två teknikerna, Chain of Thought samt Few-Shot. Respondenterna är inte direkt medvetna om att de använder en specifik teknik förrän de fått en förklaring om vad teknikerna innebär. Då visade det sig att de indirekt använt sig av teknikerna. R1 jämfört med de övriga

respondenterna har använt sig av Chain of Thought samt Few-Shot i lite större utsträckning och visar sig ha lite mer förståelse för dem.

5 Diskussion

I denna del kommer Empirin från alla intervjuer diskuteras och vävas ihop med den litteratur som tidigare studerats. Med dessa två delar kommer det att utforskas hur studenter inom informatik använder ChatGPT samt hur de använder sig av prompt engineering. Samtliga av respondenterna använde ChatGPT när de programmerade flitigt och såg det som ett kraftfullt verktyg. Däremot var det betydande skillnader bland respondenterna vad det gällde kunskap om Prompt Engineering och diverse prompt tekniker. Då alla respondenter är studenter inom området informatik kan resultatet generaliseras över de flesta studenterna inom informatik till viss mån enligt Lee och Baskerville (2003).

5.1 Respondenternas olika tekniska bakgrund

5.1.1 Likheter och skillnader

Utifrån vad resultatet säger angående respondenternas tekniska och akademiska bakgrund finns det likheter men också skillnader i deras förmåga att programmera. Likheterna är att samtliga respondenter påbörjade sina studier på det systemvetenskapliga kandidatprogrammet. Det var då de också började programmera. Några av respondenterna håller på med programmering på fritiden men den största skillnaden som visas är att respondent R1 och R2 har engagerat sig som labbhandledare vid sidan av sina studier. R1 och R2 uttrycker båda att de ökat sin kunskap inom programmering i och med att de jobbat som labbhandledare. Detta borde teoretiskt sett möjliggöra för dem att använda tekniska AI-verktyg såsom ChatGPT inom programmering bättre än övriga respondenter. Men det visar också på att de har ett större engagemang och tagit åt sig ny kunskap utanför utbildningen. Att de har ett större engagemang kan eventuellt visa på att de vågar ta sig an nya problem och anpassa sig effektivt för att lösa de problem som uppstår.

Cain (2024) menar på att ChatGPT är användarvänligt och kan användas oberoende av vilken akademisk nivå användaren besitter. Så själva användningen av ChatGPT är inte direkt kopplat med den tekniska bakgrunden användaren besitter. Detta leder i så fall till att samtliga respondenter har samma utgångsläge när de använder ChatGPT även om vissa respondenter har en större teknisk bakgrund inom programmering. I studien Wang et al. (2024) förklarar de att när en uppgift ska lösas inom programmering, med hjälp av ChatGPT, är det viktigt att man har kunskap inom både ämnet som uppgiften handlar om samt prompt engineering. Detta leder till att kvalitén på de svar ChatGPT ger har bättre kvalitet. Till skillnad från att bara använda sig av ChatGPT och faktiskt få bra kvalitet på de svar den ger sägs det nu vara viktigt enligt Wang et al. (2024), att ha förkunskap inom ämnet. Detta borde isåfall resultera i att användningen av ChatGPT inom programmering är mer gynnsamt för respondent R1 samt R2 då de besitter en något mer kunskap inom programmering eftersom de jobbat som labbhandledare till skillnad från respondent R3 och R4.

Det är en intressant skillnad på att bara använda ChatGPT och faktiskt få bra svar från den när den används inom ett specifikt ämnesområde, i det här fallet programmering. Respondenternas tekniska och akademiska bakgrund inom programmering borde enligt Wang et al. (2024) alltså spela en stor roll i hur de använder ChatGPT för att lösa olika programmeringsproblem. Men den generella användningen av ChatGPT borde, enligt Cain (2024), inte vara direkt kopplat med användarens tekniska och akademiska bakgrund. Med avseende på att lösa programmeringsproblem med hjälp av ChatGPT kan det vara så att användningen av den kan se olika ut beroende på den tekniska bakgrunden och förförståelsen inom programmering. Enligt resultatet har respondenten R1 och R2 en något högre kompetens inom programmering jämfört med respondenten R3 och R4. Detta borde i så fall leda till att användningen av ChatGPT för R1 och R2 skiljer sig åt jämfört med R3 och R4. Hur samtliga respondenter använder ChatGPT inom programmering kommer att diskuteras nedan.

5.2 Respondenternas användning av ChatGPT inom programmering

5.2.1 Olika användningsområden med ChatGPT

Utifrån resultatet identifierades olika områden där respondenterna använder ChatGPT inom programmering. De områden som samtliga respondenterna uttryckte sig använda ChatGPT till var att generera monoton kod och felsökning. Utöver detta beskrev respondenterna R2 och R4 att det också är ett bra verktyg för att lära sig nya saker inom programmering. Hutchinson (2023) nämner sex stycken olika områden där ChatGPT kan användas som ett verktyg. Två av de områden som nämns där är bland annat, felsökning (”debugging”) och att ChatGPT kan agera som en lärare och lära användaren t.ex. ett nytt programmeringsspråk. Även Cain (2024) beskriver att ChatGPT kan användas för inlärning.

Dessa två områden är precis vad respondenterna uttryckt sig använda ChatGPT för också. Att det var just de två områden som respondenterna beskrev använda ChatGPT till är kanske inte så konstigt med tanke på att de är studenter. Att vara student handlar om att man ska lära sig nya saker, därav är det naturligt att ChatGPT används som ett verktyg för att lära sig till exempel ett nytt programmeringsspråk. Även felsökning kan vara en stor del av en students användande med ChatGPT eftersom en del av att lära sig ett nytt programmeringsspråk kan handla om att felsöka den kod som inte fungerar. Som student kan det vara så att det förekommer fler fel i koden än en erfaren utvecklare. Därav är det inte så konstigt att samtliga respondenter uttryckte att de använder ChatGPT för felsökning när de programmerar.

Ett område som inte nämns i litteraturen men som samtliga respondenter beskrev att de använder ChatGPT till när de programmerar är att generera monoton kod. Till exempel beskriver Respondent R1 detta enligt följande citat:

> ”*Jag tycker att ChatGPT är bäst på att automatisera de monotona stegen i utvecklingen. Typ att generera en basklass*”,

R3 enligt följande citat:

> "Säg *att du ska göra 10 metoder i en klass och de här metoderna ska se i princip likadana ut i fem andra klasser. Att kunna automatisera, det är den bra på, jag slipper skriva det själv*"

och R4 enligt följande citat:

> ”*Mycket så här om man ska lägga in massa exempelobjekt till en databas så vill jag generera massa för att det är skönt att slippa göra själv*”.

Det verkar enligt respondenterna att detta är något som är en stor del av deras användning av ChatGPT när de programmerar. Respondent R1 säger att ChatGPT är ”*bäst*” på just detta och R2 säger ”*det är den bra på*”. En anledning till att detta område används av samtliga respondenter kan vara att en student vill vara så effektiv som möjligt och lägga tid på det som är viktigare. Att använda ChatGPT för detta område kan också leda till att man sparar tid. Det respondent R3 säger om att ChatGPT kan generera metoder som ser i princip likadana ut borde i sin tur leda till att mycket tid sparas. Das & J.V. (2024) beskriver just detta, att använda ChatGPT kan bli tidsbesparande för en student. Även om Das & J.V. (2024) inte uttryckligen pratar om detta gällande programmering kan det antas att tidsbesparande är generellt och kan appliceras på flera olika ämnen, och i detta fall programmering.

5.2.2 Förhållandet mellan teknisk bakgrund och användningen av ChatGPT

Under punkt 5.1 diskuterades respondenternas tekniska bakgrunder och vilka likheter samt skillnader de har. Det nämns också där i studien Wang et al. (2024) att kunskap inom det ämne som en uppgift berör tillsammans och prompt engineering är viktigt för att få svar från ChatGPT som har bra kvalitet. Det nämns då att den tekniska bakgrund inom programmering som respondenten R1 och R2 har jämfört med R3 och R4 isåfall borde leda till att de använder ChatGPT på ett sätt som gör att kvalitén på svaren är bättre. Men hur ser egentligen respondenternas användning av ChatGPT inom programmering ut? Resultatet visade att samtliga respondenter har ett liknande tillvägagångssätt när de använder ChatGPT inom programmering. Det finns ingen tydlig skillnad mellan de respondenter som har lite mer teknisk erfarenhet än de som inte har det.

I studien Wang et al. (2024) fann de att skillnaden i kvalitén på de svar ChatGPT gav studenterna före och efter de fått en genomgång inom området TCP inte var avsevärt stor. Resultaten i den här studien stämmer i det fallet överens med resultaten från Wang et al. (2024). Att enbart besitta en teknisk kunskap inom exempelvis programmering kommer inte märkbart att öka kvalitén på de svar ChatGPT ger. Anledning till samtliga respondenter har liknande tillvägagångssätt när de använder ChatGPT för programmering kan då vara eftersom respondenterna har liknande grundläggande förståelse för prompt engineering (diskuteras mer under punkt 5.3) men varierande teknisk bakgrund. Det är då kunskapen om prompt engineering som blir avgörande för hur användningen av ChatGPT ser ut. Därav ser respondenternas användning av ChatGPT likartad ut.

5.3 Respondenternas kunskap och användning av prompt engineering

5.3.1 Kännedom av prompt engineering

Att prompt engineering kan hjälpa LLMs såsom ChatGPT lösa programmeringsproblem till en högre grad visar både litteraturen (Denny et al. 2023) och empirianalysen på. Samtliga respondenter har en grundläggande förståelse av prompt engineering, dock har vissa bättre förståelse än andra. R1 har samlat på sig kunskap om prompt engineering på egen fot, samt har fått lära sig ännu mer av lärare på universitetet. De andra respondenterna har inte aktivt sökt efter mer kunskap inom programmering men samtliga respondenter är överens om att välskrivna prompts kan hjälpa LLMs att ge bättre output på det specifika problemet. Som Prompt Engineering Guide (2024) förklarar i litteraturavsnittet ska prompter vara simpla, effektiva och tydliga. I empirin har respondenterna svarat lite annorlunda, vilket kan tyda på en ofullständig förståelse av prompt engineering. R1 går in på att kvaliteten är viktig på prompts men utvecklar inte vidare. R2 går in på att det är viktigt för en prompt att vara koncis men samtidigt med mycket information. R3 förklarar att man ska specificera så mycket information man kan i en prompt och R4 nämner “nyckelvariabler”, vilket kan tolkas som relevant information, är viktig i prompts. Det gemensamma svar som alla respondenterna hade var att kontext i prompten är viktigt. De tar alltså upp liknande begrepp som litteraturen men de har svårt att utveckla sina svar, vilket kan tyda på bristfällig kunskap inom ämnet. Det intressanta som skulle kunna diskuteras i nästa punkt är hur deras kännedom och kunskap påverkar deras användning av prompt engineering.

5.3.2 Användning av prompt engineering

Eftersom samtliga respondenter använder sig av ChatGPT när de programmerar använder de sig av en viss form av prompt engineering även fast de inte tänker på det. I 5.3.1 blev det klart att respondenterna har en grundläggande förståelse för prompt engineering men har inte full förståelse inom ämnet. I detta avsnitt diskuteras det hur respondenterna använder sig av prompt engineering och om deras användning av prompt engineering har någon koppling till deras förståelse för ämnet.

Resultatet visar på att respondenterna i första hand skriver prompts som inte kräver mycket tid att skriva. Det kan vara till exempel att de bara skickar in felkoder till ChatGPT i hopp om att AIn ska förstå och lösa problemet. R2 försöker lösa mer komplexa problem genom att initialt skicka in en kort prompt i hopp om att få ett bra resultat. R2 menar dock att outputen på prompten misslyckas med att hjälpa lösa uppgiften. Därefter inser R2 att det behövs mer kontext och skriver en mer utformad prompt. R4 jobbar på samma sätt som R2, de värderar att prompts först ska ta kort tid att skriva, men inser att ju mer kontext och information man ger ChatGPT desto bättre output. R3 beskriver, i enlighet med R2 och R4, att kontexten är avgörande i en prompt när man programmerar. Som R2 beskriver kan det vara svårt att ge mycket kontext när man inte har mycket insikt i det programmeringsproblem man arbetar med. Detta betyder att respondenternas kvalité på prompts kan vara begränsade till deras kunskap om det specifika programmeringsproblemet.

Sammanfattningsvis kan man se att respondenterna använder sig av prompt engineering till viss grad. I enlighet med (Denny et al. 2023) inser de att kvalitén på prompts och metoder av

prompt engineering kan påverka output från LLMs som ChatGPT. Intressant nog tyder resultatet på att respondenterna värderar till stor del att prompten ska gå snabbt att skriva.

5.3.3 Kännedom och användning av prompttekniker

För att utveckla en bättre förståelse av respondenternas kunskap inom prompt engineering diskuterades specifika prompttekniker. I litteraturgenomgången valdes prompttekniker Zero-shot-prompting, Few-shot-prompting och Chain-of-thought (Wei et al, 2022;Wei et al, 2023; Min et al, 2022; Prompt Engineering Guide, 2024). Dessa tekniker valdes på grund av deras relevans inom ämnet och eventuella användningsområden inom programmering.

Resultatet visar på att samtliga respondenter hade lite eller ingen kunskap om dessa tekniker. R2 hade lite förkunskap om Chain-of-thought innebar men resten av respondenterna hade inte hört talas om de nämnda teknikerna. Det innebar att i intervjuerna förklarades de tre teknikerna för att ge insikt till respondenterna om hur de fungerade. Som Min et al. (2022) förklarar handlar Zero-shot prompting om att skicka in beskrivning av problemet utan några exempel, vilket är effektivt på mindre komplexa problem (Prompt Engineering Guide, 2024). Resultatet visar på att respondenterna använder sig av denna teknik mest. R4 förklarar att Zero-shot är tekniken som används mest, mycket för att den tar kortast tid att forma. De andra respondenternas svar tyder på att Zero-shot är den teknik de använder mest även fast de inte specificerar det ordagrant.

Few-shot prompting användes sällan av respondenterna enligt resultatet. R2 menade att tekniken har använts omedvetet. R3 och R4 förstod inte tillräckligt hur tekniken fungerade men gav exempel på hur kursmaterial skickades med i prompten. R1 gav ett intressant svar där exempel gavs på vilken struktur på output ChatGPT skulle ge sitt svar. Resultaten visar i sin helhet att respondenterna inte använder sig i någon större utsträckning av Few-shot prompting vilket kan tyda på att det kan vara svårt att applicera tekniken inom programmering.

Chain-of-thought, utvecklad av Wei et al. (2023) hjälper LLMs att bryta ner större problem till mindre och utveckla ett “step-by-step” tankesätt, detta menar författarna speciellt i mer komplexa problem. I enlighet med litteraturen visar resultatet att respondenterna använder denna teknik för just mer komplexa problem, även fast majoriteten inte kände till tekniken. R1 har använt tekniken och menar på att det är större sannolikhet att ChatGPT ger bättre svar och inte “hallucinerar” när ber den att bryta upp det i mindre steg. R1 lyfter också fram att tekniken hjälper också utvecklaren att förstå problemet bättre. R3 och R4, likt som R1, tycker att tekniken ger en output som är lättare att förstå vilket hjälper till att lösa problemet. Till skillnad från de andra använde sig R2 av tekniken i felsökning. Generellt sett kan man se att alla respondenter har använt sig av Chain-of-thought. De använder tekniken i större, mer komplexa problem precis som litteraturen säger och intressant nog nämner samtliga respondenter att tekniken hjälper dem första det stora problemet genom att bryta ner till mindre problem.

5.4 Kritiska blickar inåt

Syftet med denna sektion är att ge läsaren en inblick i potentiella förbättringsområden i studien. Transparens i forskningen är avgörande för att identifiera eventuella svagheter, vilket

möjliggör en kritisk granskning av arbetet. Samtidigt erbjuder det andra forskare inom området möjligheten att vidareutveckla sina egna studier inom ämnet.

5.4.1 Tillvägagångssätt

Eftersom denna studie fokuserar på hur informatikstudenter använder ChatGPT inom programmering var det naturligt att använda sig av intervjuer av informatikstudenter. Detta gav oss en inblick i hur studenter använder sig av ChatGPT samt i vilken omfattning. Enligt Oates et al. (2022) kan det vara fördelaktigt att intervjua systemutvecklare som arbetar på företag. Detta kan ge en bättre inblick hur världen inom informatik utanför universitetet fungerar. I denna studie skulle det alltså kunna varit fördelaktigt att intervjua en tidigare informatikstudent för att ett större perspektiv på ämnet.

5.4.2 Analys av Empiriskt material

Studier med kvalitativ metodik kan generera en omfattande mängd data, och det är forskarna själva som avgör vilken information som är relevant för studien. Enligt Oates et al. (2022) kan data kategoriseras i tre huvudsakliga grupper: data som bidrar till slutsatser, data som inte är användbar, samt deskriptiv information om forskningen. I denna studie har data sorterats genom färgkodning och en tydlig indelning i användbar respektive icke-användbar data. Den data som inte bedömdes som användbar kodades inte. Eftersom forskarna har ansvarat för denna uppdelning finns en risk för partiskhet, då önskan om ett intressant resultat kan påverka urvalet, vilket i sin tur kan resultera i att studiens resultat inte korrekt återspeglar verkligheten. För att motverka denna utmaning kan en kombination av kvalitativa och kvantitativa metoder användas, där kvantitativa data kompletterar intervjuerna och bidrar till en mer nyanserad analys.

6 Slutsats

Denna uppsats undersöker hur informatikstudenter använder sig av ChatGPT för programmeringsuppgifter och vilka tekniker de använder när de skapar sina prompts. Studien syftar till att svara på forskningsfrågan:

Hur använder informatikstudenter ChatGPT och vilka strategier använder de för att formulera effektiva prompts?

För att besvara denna fråga genomförde författarna kvalitativa intervjuer med informatikstudenter som regelbundet använder ChatGPT. Studien samlade in data om studenternas uppfattningar, erfarenheter och metoder kring ChatGPT och prompt engineering.

Resultaten visade att många studenter ser ChatGPT som ett värdefullt verktyg för att snabbt få hjälp med kodningsproblem. De fann att kvaliteten på de svar som genereras av modellen varierar beroende på hur frågorna (prompts) formuleras. Därför utvecklade många studenter sina egna strategier för att skapa effektiva prompts, såsom att specificera tydliga frågor, använda kodexempel och inkludera relevant kontext för bättre svar. Resultatet visar på att studenterna omedvetet använde sig av prompttekniker, speciellt tekniken Chain-of-thought. Vidare identifierades några olika områden som studenterna använder ChatGPT till inom programmering. Dessa områden är felsökning, generering av monoton kod samt att den agerar som lärare åt studenten.

Studien identifierar också några av de utmaningar som studenterna står inför när de arbetar med ChatGPT. Dessa utmaningar inkluderar att förstå modellens begränsningar, hantera oprecisa eller felaktiga svar och anpassa sina prompts efter modellen. Trots detta är de flesta studenter överens om att med rätt tillvägagångssätt kan ChatGPT bidra till att förbättra produktiviteten och kvaliteten på deras arbete.

Slutsatsen är att informatikstudenter aktivt använder ChatGPT för att lösa programmeringsproblem och ser stor potential i verktyget. Samtidigt betonar studien vikten av att förbättra studenternas förståelse för hur prompts bäst ska formuleras för att optimera svarens kvalitet.

6.1 Vidare forskning

Användningen av stora språkmodeller inom programmering är ett relativt nytt forskningsområde, eftersom dessa verktygs kapacitet att generera text och kod har förbättrats avsevärt under de senaste två åren. Denna studie fokuserar uteslutande på hur studenter använder ChatGPT inom programmering, vilket ger en insikt i dess användning på universitetsnivå. För att ytterligare belysa detta framväxande och fascinerande område bör framtida forskning undersöka hur yrkesverksamma utvecklare använder ChatGPT i arbetsmiljön. Detta skulle ge ett mer omfattande perspektiv på vilken inverkan LLMs som ChatGPT har på yrkesutvecklares arbete.

Referenser

Cain, W. (2024) ‘Prompting Change: Exploring Prompt Engineering in Large Language Model AI and Its Potential to Transform Education’, TechTrends: Linking Research and Practice to Improve Learning, 68(1), pp. 47–57. doi:10.1007/s11528-023-00896-0.

Das, S.R. & J.V., M. (2024). Perceptions of higher education students towards ChatGPT usage. International Journal of Technology in Education (IJTE), 7(1), 86-106. https://doi.org/10.46328/ijte.583

Denny, P., Kumar, V. & Giacaman, N. (2023) ‘Conversing with copilot: Exploring prompt engineering for solving CS1 problems using natural language’, Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1 [Preprint]. doi:10.1145/3545945.3569823.

Gartner. (2023) Generative AI: What Is It, Tools, Models, Applications and Use Cases Tillgänglig online: https://www.gartner.com/en/topics/generative-ai [Accessed: 4 May 2024]

Gartner. (2024) Gartner Says 75% of Enterprise Software Engineers Will Use AI Code Assistants by 2028 Tillgänglig online: https://www.gartner.com/en/newsroom/press-releases/2024-04-11-gartner-says-75-percent-of-enterprise-software-engineers-will-use-ai-code-assistants-by-2028 [Hämtad: 22 April 2024]

Gemini Team Google (2023) ‘Gemini: A Family of Highly Capable Multimodal Models’. arXiv. Tillgänglig online: https://doi.org/10.48550/arXiv.2312.11805

Hughes, A. (2023) ChatGPT: Everything you need to know about OpenAI’s GPT-4 tool. Tillgänglig online: https://www.sciencefocus.com/future-technology/gpt-3 [Hämtad: 4 April 2024]

Hutchinson, K. (2023) CHATGPT Guide for college students and coders, eLearning Industry. Tillgänglig online: https://elearningindustry.com/chatgpt-guide-for-college-students-and-coders#:~:text=Students%20and%20developers%20can%20use,execute%20code%20in%20those%20languages. [Hämtad: 27 Mars 2024]

Jacobsen, D.I. (2017) *Hur genomför man undersökningar? Introduktion till samhällsvetenskapliga metoder*. 2nd edn. Studentlitteratur AB, Lund.

Jacobsen, DI & Hellström, C (2002), *Vad, hur och varför: Om metodval i företagsekonomi och andra samhällsvetenskapliga ämnen*. translated by Gunnar Sandin, Studentlitteratur AB, Lund.

Kalliamvakou, E. (2022). ‘Research: quantifying GitHub Copilot’s impact on developer productivity and happiness’, The GitHub Blog, 7 September. https://github.blog/2022-09-07-research-quantifying-github-copilots-impact-on-developer-productivity-and-happiness/ [Hämtad: 19 March 2024]

Lee, A.S. & Baskerville, R.L. (2003) 'Generalizing Generalizability in Information Systems Research', Information Systems Research, 14(3), pp. 221-243. Tillgänglig online: https://www.jstor.org/stable/23015711 [Hämtad: 5 April 2024]

Leung, M. & Murphy, G. (2023) 'On Automated Assistants for Software Development: The Role of LLMs', in 2023 38th IEEE/ACM International Conference on Automated Software Engineering (ASE). 2023 38th IEEE/ACM International Conference on Automated Software Engineering (ASE), Luxembourg, Luxembourg: IEEE, pp. 1737–1741. Tillgänglig online: https://doi.org/10.1109/ASE56229.2023.00035.

Min, S., Lyu, X., Holtzman, A., Artetxe, M., Lewis, M., Hajishirzi, H., & Zettlemoyer, L. (2022) 'Rethinking the Role of Demonstrations: What Makes In-Context Learning Work?' arXiv. Tillgänglig online: http://arxiv.org/abs/2202.12837 [Hämtad: 27 Mars 2024]

Oates, B.J., Griffiths, M. and McLean, R. (2022) Researching Information Systems and Computing. SAGE.

OpenAI Platform (n.d.), Prompt Engineering, https://platform.openai.com/docs/guides/prompt-engineering/strategy-write-clear-instructions [Hämtad: 17 Mars 2022]

OpenAI. (2023). 'GPT-4 Technical Report', http://arxiv.org/abs/2303.08774 [Hämtad: 19 Mars 2024]

Prompt Engineering Guide (2024). Tillgänglig online: https://www.promptingguide.ai/introduction/tips [Hämtad: 26 Mars 2024]

Rodriguez, A.D., Dearstyne, K.R. & Cleland-Huang, J. (2023) 'Prompts Matter: Insights and Strategies for Prompt Engineering in Automated Software Traceability', 2023 IEEE 31st International Requirements Engineering Conference Workshops (REW), Requirements Engineering Conference Workshops (REW), 2023 IEEE 31st International, REW, pp. 455–464. doi:10.1109/REW57809.2023.00087.

Sahoo, P., Singh, A., Saha, S., Jain, V., Mondal, S., & Chadha, A. (2024) 'A Systematic Survey of Prompt Engineering in Large Language Models: Techniques and Applications', http://arxiv.org/abs/2402.07927 [Hämtad: 19 Mars 2024]

Velásquez-Henao, J.D., Franco-Cardona, C.J. & Cadavid-Higuita, L. (2023) 'Prompt Engineering: a methodology for optimizing interactions with AI-Language Models in the field of engineering', Dyna, 90(230). doi:10.15446/dyna.v90n230.111700.

Wang, M., Wang, M., Xu, X., Yang, L., Cai, D., & Yin, M (2024) 'Unleashing ChatGPT's Power: A Case Study on Optimizing Information Retrieval in Flipped Classrooms via Prompt Engineering', IEEE Transactions on Learning Technologies, 17, pp. 629–641. doi:10.1109/TLT.2023.3324714.

Wei, J., Bosma, M., Zhao, V., Guu, K., Yu, A., Lester, B., Du, N., Dai, A., & Le, Q. (2022) 'Finetuned Language Models Are Zero-Shot Learners'. arXiv. Tillgänglig online: http://arxiv.org/abs/2109.01652 [Hämtad: 26 Mars 2024]

Wei, J., Wang, X., Schuurmans, D., Bosma, M., Ichter, B., Xia, F., Chi, E., Le, Q., & Zhou, D. (2023) ‘Chain-of-Thought Prompting Elicits Reasoning in Large Language Models’. arXiv. Tillgänglig online: http://arxiv.org/abs/2201.11903 [Hämtad: 27 Mars 2024]