

UPPSALA
UNIVERSITET

UPTEC STS 25028

Examensarbete 30 hp

Juni, 2025

Generering av kunskapsgrafer med hjälp av stora språkmodeller

En automatiserad kartläggningsprocess av kursmaterial på lärandeplattformen Kollin

Samuel Ram



Civilingenjörsprogrammet i System i Teknik och Samhälle



UPPSALA
UNIVERSITET

Generering av kunskapsgrafer med hjälp av stora språkmodeller

Samuel Ram

Abstract

This thesis investigates how Large Language Models (LLMs) can be utilized to automatically generate knowledge graphs from mathematical theory chapters stored in Kollin's database. Two prompting techniques were evaluated - *stepwise prompting*, where all subtasks are executed in one single LLM call, and *prompt chaining*, where a sequence of LLM calls tackle one subtask each. To create the prompts for both prompting techniques, a method called *Recursive Meta Prompting* was used - where an LLM iteratively creates and updates the prompt with the help of human input. Both techniques produced coherent graphs, but due to the inherent stochasticity in LLMs, the results varied from graph to graph. However, prompt chaining offered superior transparency and debuggability at the cost of multiple LLM calls. Furthermore, it was investigated to which degree adjusting the hyperparameter settings of the LLM had a significant effect on output stability in the keyword extraction step of the prompt chaining sequence. A temperature setting of $t \leq 0.3$ ensured stable keyword extraction, while variations in top-k and top-p did not have a significant impact. Varying the level of detail in the system prompts revealed that both minimal and highly detailed instructions led to lower tolerance for temperature changes, whereas a moderate level of detail allowed for greater robustness to such variations. Qualitative analysis of selected chapters confirmed that using a higher temperature on occasion included keywords with less relevance, but also showed that some of the variation in the output could be explained from the LLM extracting different grammatical variants of the same concept. Future studies should focus on incorporating *Retrieval-Augmented Generation (RAG)* to surpass context window constraints (token limits), and on hybrid solutions that combine automation with expert-crafted ontologies.

Teknisk-naturvetenskapliga fakulteten

Uppsala universitet, Utgivningsort Uppsala

Handledare: Kristofer Espinosa Ämnesgranskare: Lars Oestreicher

Examinator: Elísabet Andrésdóttir

Sammanfattning

Denna masteruppsats undersöker hur stora språkmodeller kan nyttjas för att automatiskt generera kunskapsgrafer från teorikapitel inom matematik lagrade i Kollins databas. Två promptingtekniker analyserades - *stepwise prompting*, där alla deluppgifter bearbetas i ett enda anrop till språkmodellen, och *prompt chaining*, där en sekvens av anrop till språkmodellen hanterar en deluppgift vardera. För att skapa prompten för båda promptingteknikerna användes metoden *Recursive Meta Prompting* - där en språkmodell används för att iterativt skapa och uppdatera promptet med hjälp av mänsklig input. Båda teknikerna kunde producera grafer, men givet att stora språkmodeller har en inneboende stokastisk egenskap, kunde resultaten variera från graf till graf. Hursom så gav prompt chaining större möjlighet till transparens och felsökning, på bekostnad av flera anrop. Vidare undersöktes det till vilken grad justering av hyperparameterinställningarna för den stora språkmodellen hade någon signifikant påverkan på utdatans stabilitet i nyckelordsextraheringssteget i prompt chaining-sekvensen. En temperaturinställning på $t \leq 0{,}3$ säkrade en stabil nyckelordsextraktion, medan variationer i top-k och top-p inte hade någon signifikant påverkan. Att variera detaljnivån i prompten visade att både minimala och detaljerade instruktioner ledde till en lägre tolerans för temperaturförändringar, medan en mer balanserad detaljnivå visade sig vara mer robust mot sådana variationer. Kvalitativ analys av utvalda kapitel bekräftade att en högre temperatur ibland inkluderade nyckelord med lägre relevans, men visade också att en del av variationen kunde förklaras av att språkmodellen extraherade olika grammatiska varianter av samma koncept. Framtida studier bör fokusera på att inkorporera *Retrieval-Augmented Generation (RAG)* för att tackla begränsningar relaterade till kontextfönstret, och på hybrida lösningar som kombinerar automatisering med ontologier skapade av domänspecifika experter.

Författarens tack Jag vill rikta ett stort tack till min handledare Kristofer Espinosa för hans vägledning och stöd under arbetets gång. Ett stort tack ska också riktas till Hampus och Sina på Kollin som alltid kommit med ovärderlig teknisk input. Slutligen vill jag även tacka min ämnesgranskare Lars Oestreicher som har följt mitt arbete med entusiasm och gett mig värdefulla perspektiv.

Samuel Ram

Innehåll

1 Introduktion **1**

1.1 Bakgrund 1

1.2 Tidigare forskning 2

1.3 Forskningsfrågor 3

1.4 Begränsningar 3

2 Litteratur & teori **4**

2.1 Stora språkmodeller 4

2.1.1 Förståelse 4

2.1.2 Grundläggande arkitektur 5

2.1.3 Tokens 8

2.1.4 Hyperparametrar 9

2.1.5 RAG 11

2.2 Promptingtekniker 11

2.2.1 Prompt chaining & Stepwise prompting 11

2.2.2 Meta prompting 12

2.3 Kunskapsgrafer 13

2.3.1 Definition 14

2.3.2 Ontologi 15

2.3.3 Tillämpningar 15

3 Metod **16**

3.1 Fas 1 - Stepwise prompting 17

3.2 Fas 2 - Prompt chaining 19

3.3 Fas 3 - Nyckelordsextrahering 21

3.4 Statistiska mått 22

3.4.1 Jaccardlikhet 22

3.4.2 Spearmankorrelation 23

3.4.3 R^2 . . . 23
3.4.4 Kruskal-Wallis-test & Dunns Post-hoc-analys . . . 23
3.5 Verktyg . . . 24
3.5.1 Claude . . . 24
3.5.2 Neo4j . . . 25
3.5.3 n8n . . . 26

4 Resultat **28**
4.1 Stepwise prompting . . . 28
4.1.1 Promptgenerering . . . 28
4.1.2 Kunskapsgrafer genererade med stepwise prompting . . . 29
4.2 Prompt chaining . . . 33
4.2.1 Promptgenerering . . . 33
4.2.2 Kunskapsgrafer genererade med prompt chaining . . . 35
4.3 Tester av nyckelordsextrahering . . . 37
4.3.1 Kvantitativ analys . . . 37
4.3.2 Kvalitativ analys . . . 45

5 Diskussion **49**
5.1 Stepwise prompting & prompt chaining . . . 49
5.2 Insikter från testerna av nyckelordsextrahering . . . 50
5.2.1 Kvantitativ analys . . . 50
5.2.2 Kvalitativ analys . . . 51
5.3 Alternativa tillvägagångssätt . . . 53

6 Slutsatser **56**
6.1 Framtida studier . . . 57

A Modellen för Prompt chaining-systemet **61**

B Pseudokod för Python-loop **62**

C Slutgiltig version av prompt för stepwise prompting **63**

D Prompts för prompt chaining **76**

D.1 Prompt som ger beskrivningar till nyckelorden 76

D.2 Prompt som skapar relationer mellan nyckelorden 76

D.3 Prompt som integrerar ett kapitel i grafen 77

D.4 Prompt som validerar hela grafen 80

E Olika promptvarianter för nyckelordsextrahering **81**

E.1 Minimal variant 81

E.2 Detaljerad variant 81

F Kapitel om Matrisinvers **83**

1 Introduktion

I takt med att artificiell intelligens (AI) och stora språkmodeller (engelska: *Large Language Models - LLMs*) utvecklas, öppnas nya möjligheter för att förbättra lärandeprocesser och skapa mer personaliserade studieupplevelser. En av dessa möjligheter är att använda AI för att generera kunskapsgrafer baserade på kursmaterial, vilket kan hjälpa studenter att identifiera och arbeta med områden de har svårt för. Kunskapsgrafer är strukturerade representationer av information som består av begrepp och relationer mellan dem, och är en datastruktur som har många olika användningsområden. Användning av kunskapsgrafer inom utbildning och lärande är ett växande forskningsområde, likaså användning av stora språkmodeller som stöd för konstruktionen av dessa.

1.1 Bakgrund

Kollin är ett företag vars syfte är att hjälpa svenska ingenjörsstudenter ta sig igenom sina studier. Den grundläggande idén för Kollins tjänst är att studenter inte ska behöva 'uppfinna hjulet' år efter år vad gäller inlärning och struktur av studier. Tjänsten är mer specifikt inriktad på studier inför tentor, och ämnar att hjälpa studenter bli så väl förberedda inför dessa som möjligt. Tjänsten täcker ett flertal kurser från olika universitet runt om i landet. Se tabell 1 för antalet kurser Kollin i dagsläget täcker.

Tabell 1: Antal kurser tillgängliga i Kollin uppdelat per universitet

Universitet	Antal kurser
Kungliga tekniska högskolan (KTH)	419
Linköpings universitet (LiU)	56
Lunds tekniska högskola (LTH)	48
Uppsala universitet (UU)	36
Chalmers tekniska högskola	27
Luleå tekniska universitet (LTU)	20
Stockholms universitet (SU)	19
Jönköpings universitet	4
Handelshögskolan i Stockholm (SSE)	2

Som kan ses i tabell 1 så är majoriteten av kurserna som finns tillgängliga på Kollin från Kungliga tekniska högskolan, vilket kan förklaras av att grundarna av företaget själva är examinerade studenter från KTH, och därmed naturligt har en starkare bas där för att samla in kursmaterial. En typisk kurssida på Kollin innehåller både teori och uppgifter. Teoriavsnitten är systematiskt indelade utefter kursområde och så är även uppgifterna. De uppgifter som är tillgängliga i Kollins kursutrymmen kommer från gamla uppladdade tentor, och anses därför vara en värdefull resurs för studenter att använda då de med största sannolikhet är av liknande karaktär som uppgifterna på kommande tentor. I och med att de uppladdade tentauppgifterna har kategoriserats och sorterats underlättar det för studenterna att mer fokuserat kunna studera kursområden de har svårt för. Uppgifterna

kan självskattas av studenterna, det vill säga att studenten får ange hur svår uppgiften var, vilket gör det möjligt att för varje uppgift se vad den sammanvägda medelskattningen är. På så sätt kan studenten välja vilken svårighetsgrad på uppgifterna hen vill ta sig an (Kollin 2025).

Kollin jobbar aktivt med att undersöka hur deras tjänst kan utvecklas och bli mer relevant för den aktuella målgruppen. Kollin kan kategoriseras som en onlinebaserad inlärningstjänst och i denna kategori går det att finna andra aktörer som utmärker sig på olika sätt. En subkategori av inlärningstjänster är MOOCs - Massive Open Online Courses. De kännetecknas av att de är öppna för alla, onlinebaserade, och 'massiva' i den bemärkelse att de kan hantera ett stort antal deltagare samtidigt. Inlärning i dessa tjänster består exempelvis av förinspelade videoföreläsningar och interaktiva quiz. I denna kategori ingår tjänster såsom *Duolingo*, *KhanAcademy* och *Coursera*. Dessa plattformar lägger mycket fokus på pedagogik och inlärningsdesign, men saknar specifikt riktat innehåll, i och med att kursinnehållet är detsamma för alla som använder tjänsterna. En annan stor subkategori är LMS - Learning Management Systems. Dessa är utbildningssystem som är till för att hantera, distribuera och följa upp lärande främst inom skolor och universitet. I ett LMS är det möjligt att hantera kurser, dela material, samla in uppgifter, sätta betyg, och kommunicera direkt mellan kursadministration och deltagare. Dessa underlättar kursorganisation både för kurser som hålls helt på distans och på plats. Till skillnad från MOOC karaktäriseras alltså LMS av en hög grad av personalisering. Exempel på LMS-tjänster är *Canvas*, *Moodle*, och *Google Classroom*.

Kollin har som mål att utvecklas inom båda de dimensioner där MOOCs och LMS har sina styrkor. Som det redan är i nuläget har Kollin en unik position i att kursmaterialet kommer från specifika kurser från svenska universitet, till skillnad från MOOC-tjänster som utvecklar eget mer 'generellt' innehåll, samtidigt som de har gjort insatser att utforma ett pedagogiskt gränssnitt där kursmaterial kan sorteras och bearbetas utifrån kategori och svårighet. För att excellera ännu mer i båda dimensionerna ser Kollin det som en möjlighet att implementera AI-stödda tjänster, något de redan till viss del har gjort med lanseringen av deras egna AI-assistent, Kollin AI, förkortat KAI. KAI är en chatbot som finns tillgänglig i Kollins gränssnitt för att bland annat generera pedagogiska lösningsförslag på tentafrågor och svara på domänspecifika frågor. Den kan även generera exempeluppgifter på olika nivåer utöver de tentauppgifter som är uppladdade i tjänsten. För att ta personaliseringen ännu längre samtidigt som en hög nivå av inlärningsdesign eftersträvas, uppkom idén om en smart studieplan. Kollin har redan i dagsläget ett fåtal färdiga studieplaner för några enskilda kurser, men dessa är designade av externa matematiska experter. Att kunna ta fram en lösning som automatiskt skulle kunna kartlägga en kurs, skapa en inlärningsväg genom denna, och samtidigt ta hänsyn till hur den enskilda studenten presterar, hade varit värdefullt för Kollin. Ur denna framtidsvision uppkom behovet av detta arbete, vars fokus ligger på själva kartläggningen av kurserna.

1.2 Tidigare forskning

Forskning om kunskapsgrafer inom utbildning har ökat under det senaste decenniet, särskilt inom högre utbildning och STEM-ämnen som matematik och datavetenskap (Qu m. fl.

2024). Samtidigt har användningen av stora språkmodeller för att skapa och komplettera kunskapsgrafer börjat utforskas. Ett exempel på detta är Abu-Rasheed m. fl. (2025), som presenterar ett semi-automatiserat arbetssätt där stora språkmodeller extraherar och klassificerar begrepp från föreläsningsmaterial i syfte att strukturera universitetskurser i en kunskapsgraf. Modellen kräver dock mänsklig validering av både begreppsindelning och semantiska relationer, vilket begränsar graden av automatisering. Samtidigt identifierar Qu m. fl. (2024) flera övergripande brister i nuvarande forskning om kunskapsgrafer i utbildning, däribland begränsad skalbarhet, svårigheter med ämnesövergripande tillämpning samt brist på automatiserade metoder för grafkonstruktion. Dessutom är användningen av stora språkmodeller för att automatiskt extrahera matematiskt ämnesinnehåll - särskilt från teoritunga kapitel - ännu inte adresserad i någon större omfattning. Sammanlagt tyder tidigare forskning på en tydlig kunskapslucka: medan stora språkmodeller har visat potential i assisterade arbetsflöden, saknas studier som systematiskt undersöker möjligheten till fullt automatiserad konstruktion av kunskapsgrafer från undervisningsmaterial. Detta utgör en viktig motivering till att undersöka hur långtgående automatisering kan uppnås med hjälp av stora språkmodeller, utan att vara beroende av manuell efterbearbetning.

1.3 Forskningsfrågor

Enligt visionerna som angetts i bakgrunden är syftet med detta arbete att undersöka huruvida betydelsefulla kunskapsgrafer kan skapas med hjälp av generativ AI i form av stora språkmodeller. För att uppfylla detta syfte har följande forskningsfrågor formulerats:

- Kan en fullt ut automatiserad process med hjälp av stora språkmodeller producera meningsfulla kunskapsgrafer baserade på kursmaterial?
- Hur påverkar val av promptlängd och inställningar av hyperparametrar stabiliteten och kvaliteten på dessa automatiskt genererade kunskapsgrafer?

1.4 Begränsningar

En begränsning i detta arbete är att det kursmaterial som använts uteslutande har rört sig om matematiska kurser på universitetsnivå. Därför bör inga slutsatser dras från resultaten angående hur dessa stora språkmodeller skulle prestera på teoritexter på annan studienivå eller annan domän. Allt kursmaterial som använts är hämtat från Kollins plattform, där allt material är skrivet på *markdown*-format, vilket innebär att de är inbäddade i HTML-taggar, vilket är en begränsning då inget material på rent textformat har använts. Dessutom har arbetet endast använt en (1) stor språkmodell, och slutsatser bör därför dras med försiktighet gällande språkmodellers generella prestanda på detta område utifrån denna studie.

2 Litteratur & teori

I detta kapitel kommer den litteratur och teori presenteras som är nödvändig för att förstå arbetet, samt ge en djupare förståelse för de tekniker som använts. Kapitlet behandlar hur stora språkmodeller fungerar, vad kunskapsgrafer är, samt forskning relaterad till olika promptingtekniker.

2.1 Stora språkmodeller

Stora språkmodeller är avancerade AI-system som är designade för att bearbeta, förstå, och generera text som efterliknar mänsklig kommunikation. De tränas på stora mängder textmaterial från en stor variation av källor såsom böcker, webbsidor, och artiklar vilket gör att de kan 'förstå' grammatik, sammanhang och nyanser. Här följer en gedigen genomgång av de fundamentala mekanismerna i stora språkmodeller.

2.1.1 Förståelse

Först och främst, innan den tekniska biten avhandlas, är det av värde att bena ut vad förståelse innebär, och till vilken grad stora språkmodeller faktiskt har någon form av mänsklig förståelse. Detta är något som Arcas (2022) ämnar att besvara. Hans artikel *Do Large Language Models Understand Us?* är visserligen endast några få år gammal, men givet att artificiell intelligens och stora språkmodeller är ett område där utveckling sker i hög takt, är det inte omöjligt att vissa av de frågor som artikeln ställer sig redan har retts ut. I artikeln skriver författaren explicit att det än inte är många som har haft möjligheten att interagera med en AI-stödd chatbot, något som inte längre stämmer. Det frångår inte att de filosofiska frågor som lyfts av Arcas (2022) fortsatt är relevanta.

Ett av argumenten som Arcas (2022) för är att han motsätter sig tanken på att artificiell intelligens och maskininlärning 'bara handlar om statistik', och att det därför inte kan leda till 'riktig' förståelse. Artikelförfattaren argumenterar att *'Statistics do amount to understanding, in any falsifiable sense.'*. Med detta menar han att om man ska hålla sig till det som går att testa rent vetenskapligt och empiriskt, kan vi inte betrakta förståelse som ett magiskt fenomen. Förståelse ska snarare betraktas som förmågan att se, känna igen, och generalisera mönster - vilket är just det som statistiska modeller gör. Därför måste vi, enligt Arcas (2022), acceptera att en modell som konsekvent beter sig som att den förstår, också *har* förståelse i viss mening. Vidare så erkänner han att förståelse inte är binärt och att förståelse för ett givet koncept kan vara allt på ett spektrum från ytlig till djup och nyanserad. Dock finns det inga väldefinierade gränser i det spektrumet och det finns därför inget tillförlitligt sätt att skilja 'falsk' förståelse från 'sann' förståelse - därför bör statistisk förståelse betraktas som en legitim form av förståelse. Enligt Arcas (2022) blir det svårt att motivera varför digitala nätverk inte skulle kunna förståelse om vi accepterar att våra egna hjärnor fungerar utifrån liknande principer - biologiska neurala nätverk som lär av erfarenhet.

2.1.2 Grundläggande arkitektur

Dagens stora språkmodeller bygger på en arkitektur som kallas transformer-baserade modeller. Transformerarkitekturen togs fram av Vaswani m. fl. (2017) och var revolutionerande inom *Natural Language Processing (NLP)* då de ersatte den dåvarande standarden *Recurrent Neural Networks (RNN)* genom att addera en uppmärksamhetsmekanism.

Recurrent Neural Networks Sherstinsky (2020) förklarar i sin artikel *Fundamentals of Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) Networks* hur föregångaren till dagens transformbaserade modeller fungerar. RNN har designats för att hantera sekventiell data - alltså data där ordningen på datapunkterna har betydelse. Till skillnad från vanliga 'feedforward'-nätverk har RNN en återkopplingsfaktor. Ett neuralt nätverk utan återkoppling lämpar sig bäst för data där ordningen på input inte har någon betydelse - ett tydligt exempel på detta skulle vara bildklassificering. Hur en bild klassificeras bör inte vara beroende av vilken bild som gavs som tidigare input. Ett exempel där fördelen med RNN blir tydlig är fallet då en modell ska förutsäga nästa ord i en mening. I ett traditionellt neuralt nätverk skulle varje ord i meningen behandlas som en oberoende input, och nätverket skulle då lära sig förutsäga nästkommande ord baserat på observerade mönster i träningsdatan. Det skulle dock vara omöjligt för nätverket att ha någon kunskap om vilket ord som föregått det aktuella ordet. I det fallet är RNN ett mer lämpligt tillvägagångssätt då nätverket, tack vare återkopplingen, kan behålla ett slags internt minne och därmed göra mer rimliga förutsägelser, då nätverket kan ta hänsyn till de ord som kommit före det ordet som utvärderas (EITCA Academy 2023).

Utan att gå in för djupt i hur ett nätverk är uppbyggt, så representeras minnet rent matematiskt av en tidsförskjutning, precis som i signal- och reglersystem (Sherstinsky 2020). En grundläggande formel för ett RNN är följande:

$$s[n] = W_s s[n-1] + W_r r[n-1] + W_x x[n] + \theta_s \tag{1}$$
$$r[n] = G(s[n]) \tag{2}$$

där

- $s[n]$ är det dolda tillståndet vid tidpunkt n
- $r[n]$ är utdata vid tidpunkt n
- $x[n]$ är input vid tidpunkt n
- W_s, W_r, W_x är viktmatriser
- θ_s är biasvektorn
- $G(\cdot)$ är en aktiveringsfunktion (t.ex. tanh)

Det är utifrån ovanstående tydligt att ett RNN är konstruerat på så sätt att det dolda tillståndet i modellen, det som också kan kallas för 'minnet', är beroende av tidigare inputs (Sherstinsky 2020).

Long Short-Term Memory (LSTM) Long Short-Term Memory (LSTM) är en särskild typ av RNN som är designad för att lösa de vanliga problem som uppstår i vanliga RNNs. Ett exempel på ett sådant problem är att RNNs tenderar att förlora/'tappa' tidigare information över lång tid, och LTSM är därmed konstruerad för att behålla viktig information över längre tid. LSTM-modellen har en minnescell, vars syfte är att behålla sitt värde över tid. I en klassisk RNN-modell är minnet känsligt i och med att det kan 'skrivas över' av nästkommande input. I LSTM uppdateras istället minnet kontrollerat med hjälp av tre stycken 'grindar' (engelska: 'gates', Hochreiter och Schmidhuber 1997):

- *Input gate* - Denna grind bestämmer hur mycket ny information som ska läggas till i minnet
- *Forget gate* - Denna grind bestämmer hur mycket av tidigare minne som ska glömmas
- *Output gate* - Denna grind styr hur mycket av tidigare minne som ska släppas ut som output

Varje grind representeras av formeln:

$$\vec{z}_{cu}[n] = W_{xcu}\vec{x}[n] + W_{scu}\vec{s}[n-1] + W_{vcu}\vec{v}[n-1] + \vec{\theta}_{cu} \tag{3}$$
$$\vec{g}_{cu}[n] = G_c(\vec{z}_{cu}[n]) \tag{4}$$

Där:

- $x[n]$ – **Inputvektor** vid tidpunkt n.
- $s[n-1]$ – **Föregående intern tillståndsvektor** (dold tillståndssignal).
- $v[n-1]$ – **Föregående utgångssignal** från cellen (readout/output).
- $W_{xcu}, W_{scu}, W_{vcu}$ – **Viktmatriser** för respektive signal.
- θ_{cu} – **Biasvektor** för update-grinden.
- $z_{cu}[n]$ – **Linjär kombination** innan aktivering.
- $g_{cu}[n]$ – **Gateoutput**, dvs. styrsignal i intervallet $[0, 1]$.
- $\sigma(\cdot)$ – **Sigmoidfunktion**, används för att begränsa gateoutputen.

Det problem som finns i RNNs med att gamla inputs snabbt glöms bort då de skrivs över av nya, löses med hjälp av grindarna i en LSTM-modell då dessa 'bestämmer' vad som ska finnas i modellens långtidsminne. Grindarna lär sig vad som är viktigt att lägga till, glömma respektive släppa ut som output genom träning då de är neuronala enheter med egna vikter (Sherstinsky 2020).

Transformbaserade modeller RNN- och LSTM-modeller var länge så kallade 'State Of The Art', det vill säga det senaste och mest moderna, fram till att Vaswani m. fl. (2017) tog fram den transformbaserade modellen på uppdrag av Google. Den huvudsakliga skillnaden mellan transformbaserade modeller och dess föregångare är förmågan att bearbeta data parallellt, till skillnad från RNN- och LSTM-modeller som bearbetar data sekventiellt. Detta möjliggörs genom så kallade självuppmärksamhetsmekanismer ('self-attention') där modellen kan väga betydelsen av varje element i en sekvens gentemot alla andra element samtidigt. Hur detta fungerar rent praktiskt i modellen är att varje position i indatan (vilket skulle kunna vara ett ord i en mening exempelvis) omvandlas till tre olika vektorer, en **query** (fråga), en **key** (nyckel) och en **value** (värde). För att avgöra hur mycket 'uppmärksamhet' en *token* (se kapitel 2.1.3) ska ge ett annat, beräknas en likhetspoäng mellan query och key för alla par i sekvensen. Poängen normaliseras sedan med en softmax-funktion och används som vikter för att kombinera value-vektorerna, som i sin tur ger en ny representation för varje position. Formeln för denna beräkning kan ges som:

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V \tag{5}$$

där Q, K och V är matriser med query, key och value-vektorer, och d_k beskriver dimensionen på key-vektorerna. Transformmodellen använder dessutom **multi-head self-attention**, som innebär att flera självuppmärksamhetsmekanismer körs parallellt med olika parametrar. Fördelen med det är att flera olika typer av relationer i sekvensen kan fångas samtidigt. Vid multi-head attention projiceras Q, K och V till h olika representationer med olika parametrar för varje *head* enligt följande:

$$head_i = Attention(QW_i^Q, KW_i^K, VW_i^V) \tag{6}$$

där W_i^Q, W_i^K, och W_i^V är projektionernas viktmatriser för head i. Alla heads beräknas parallellt och resultaten räknas samman och projiceras med en viktmatris W^O:

$$MultiHead(Q, K, V) = Concat(head_1, ..., head_h)W^O \tag{7}$$

Multihead-mekanismen gör att flera olika mönster och relationer i datan kan uppmärksammas samtidigt, medan single-head endast har kapacitet att fokusera på ett 'perspektiv' åt gången. Arkitekturen i transformmodellen har lett till stora framsteg inom språkmodellering. Ett exempel på modell som använder transformarkitekturen är BERT (*Bidirectional Encoder Representations from Transformers*) som kännetecknas av **bidirektionell kontextuell inlärning**, vilket innebär att den tar hänsyn till både vänster och höger kontext vid varje token, det vill säga den omgivande texten till vänster och höger om ett ord i en textsekvens, medan tidigare modeller hade varit ensidiga och bara tagit hänsyn till de föregående orden eller de efterföljande, men inte båda samtidigt (Devlin m. fl. 2019).

En annan språkmodell baserad på transformerarkitektur är GPT, som står för *Generative Pre-trained Transformer*. Som beskrivs i Brown m. fl. (2020), är transformerarkitekturen i GPT autoregressiv. Det innebär att modellen förutsäger nästa ord i en sekvens, givet tidigare ord. GPT använder en variant av self-attention som kallas 'casual self-attention' vilket innebär att varje position bara får tillgång till tidigare positioner i sekvensen, till skillnad från BERT som ger tillgång både till tidigare och följande sekvenser (Brown m. fl. 2020).

I transformerarkitekturen som både GPT och BERT bygger på, finns det två huvudsakliga komponenter som kallas för **encoder** och **decoder**. Encoder är den del av arkitekturen som används för att förstå, tolka och representera en inputsekvens. Decoder är den del som används för att generera text. BERT är en så kallad 'encoder-only', där modellens huvudsakliga syfte är att tränas till djup språkförståelse (Devlin m. fl. 2019). I kontrast står GPT som är en 'decoder-only', vars huvudsakliga syfte är att generera sammanhängande och naturlig text. Den ursprungliga transformermodellen som presenterades av Vaswani m. fl. (2017) har både en encoderdel och decoderdel. Ett användarfall när detta kan behövas är exempelvis i maskinöversättning, då en encoder förstår en text på ett språk och skapar en vektorrepresentation av den, varpå decodern använder den vektor-inputen för att generera text av motsvarande betydelse på ett annat språk (Vaswani m. fl. 2017).

2.1.3 Tokens

En *token* är en grundläggande enhet för textrepresentation i stora språkmodeller. En token kan vara ett ord, teckenuppsättningar eller kombinationer av ord och skiljetecken som genereras av stora språkmodeller när de behandlar och delar upp text. När en modell tränas är tokenisering ofta det första steget, där språkmodellen får analysera semantiska relationer mellan tokens, såsom hur ofta de förekommer tillsammans eller om de förekommer i liknande kontexter. Det är dessa mönster och relationer som sedan ligger som grund till den sekvens som språkmodellen genererar som output (Microsoft 2024). Det finns olika tokeniseringsmetoder, där några vanliga är:

- **Word-tokenisering** - text delas upp i enskilda ord
- **Teckentokenisering** - text delas upp i enskilda tecken
- **Underordstokenisering** - text delas upp i delar av ord eller teckenuppsättningar

Stora språkmodeller har begränsningar gällande det maximala antalet tokens som kan ges som input eller genereras som output. Denna begränsning gör ofta att input- och outputtoken kombineras till ett maximalt kontextfönster. Ponera att det maximala kontextfönstret är 100 tokens och inputtexten lyder:

```
Jag ska gå och handla mat
```

Om en ordbaserad tokeniseringsmetod har använts är input 6 tokens, så detta lämnar 91 ordtokens för output, men om en teckenbaserad tokeniseringmetod använts är inputen

24 tokens, och endast 74 teckentokens kan ges i outputen. Studien *Same task, more tokens* av Levy, Jacoby och Goldberg (2024) visar dock att språkmodellers prestanda starkt påverkas av att antalet inputtokens ökar - trots att de mycket väl kan hålla sig inom ramen för kontextfönstret. De slår fast att stöd för längre indata inte automatiskt innebär att modellen kan leverera ett robust resonemang som utdata. Utöver det belyser även en artikel av D. Wang m. fl. (2024) att tokenisering är ett förbättringsområde inom utveckling av stora språkmodeller. De lyfter bland annat det faktum att mänsklig förståelse och tokenisering i stora språkmodeller ofta kan skilja sig åt. Exempelvis skulle sekvensen:

```
he moves table
```

tolkas av en människa som 'he moves table' vilket vi skulle 'tokenisera' till → 'move' och 'table', medan en stor språkmodell lika gärna skulle kunna tolka 'he movestable' vilket skulle ge tokeniseringen → 'move' och 'stable'. I en hyperparameterkontext går det att ange hur många tokens som outputen maximalt får bestå av.

2.1.4 Hyperparametrar

De flesta moderna stora språkmodeller har ett antal olika hyperparametrar som kan ställas in för att styra hur outputen ska genereras. Tre vanligt förekommande hyperparametrar är **temperature**, **top-k**, och **top-p**.

Temperature *Sampling temperature* är en hyperparameter som i grund och botten kontrollerar nivån av slumpmässighet i modellens output. När modellen genererar text, beräknas sannolikheten för nästa token med hjälp av tidigare tokens. Justering av temperaturen gör att modellen får en anpassad sannolikhetsfördelning och på så sätt kan bli mer eller mindre slumpmässig (Renze 2024). Temperatur används på följande sätt i stora språkmodeller:

$$Pr(v_k) = \frac{e^{l_k/\tau}}{\sum_i e^{l_i/\tau}} \tag{8}$$

Där:

- $Pr(v_k)$ - sannolikheten att modellen väljer token v_k som nästa ord i sekvensen
- l_k - *logiten*, alltså modellens 'poäng' för v_k innan en softmax-funktion omvandlar den till andel
- τ - samplingstemperaturen

Det ekvation 8 beräknar är alltså hur troligt det är att modellen väljer just token v_k givet alla de andra möjliga alternativen, och temperaturen τ används för att kontrollera formen på sannolikhetsfördelningen. När $\tau \to 0$ resulterar det i att sannolikheten går mot 1 för

den mest sannolika token, och går mot 0 för alla andra. Alltså blir outputen mer deterministisk. På samma sätt kommer sannolikheten för alla tokens bli identiska när $\tau \to \infty$, vilket gör outputen helt slumpmässig. I praktiken används 1 som en övre gräns för temperaturen, och inte ∞. Vilka värden på temperaturen som bör användas beror på vad syftet med uppgiften i fråga är. Låga temperaturer rekommenderas för uppgifter som kräver precision och faktamässig noggrannhet såsom tekniska texter och kodbaserade uppgifter , medan höga temperaturer rekommenderas för uppgifter som kräver ett högre mått av kreativitet, såsom poesiförfattande och brainstorming. I sin artikel om just sampling temperature påpekar dock Renze (2024) att de flesta riktlinjer kring detta som finns i 'prompt engineering-communityt' ofta är vaga och anekdotbaserade, och att de saknar stöd i empiriska studier. De menar därför att hur man väljer en optimal samplingtemperatur för ett specifikt problem för det mesta är baserat på gissningar, magkänsla och icke-systematiskt experimenterande (Renze 2024).

Top-k Top-k är en hyperparameter som styr hur många alternativ som övervägs under samplingen. Efter att temperaturen har styrt sannolikhetsfördelningen, och sannolikheterna är beräknade, kan det fortfarande finnas tusentals tokens att välja mellan. Genom att specificera ett värde för top-k, ger man ett värde för hur många av dessa tokens modellen ska överväga att välja som nästa output. Om top-k sätts till 50 exempelvis, är det som att säga till modellen: 'Överväg endast de 50 bästa tokens' (Hyrkas 2023). Holtzman m. fl. (2019) lyfter att det kan finnas svårigheter med att välja detta värde då det inte alltid är optimalt att använda ett konstant top-k i alla sammanhang. I vissa situationer är sannolikhetsfördelningen för nästa ord ganska jämn, såsom för substantiv eller verb i generiska sammanhang, medan största delen av sannolikheten i andra situationer är koncentrerad till ett eller några få ord. Om k är litet riskerar man att få en 'tråkig' eller allt för generell text, medan ett stort k kan leda till att olämpliga ord inkluderas (Holtzman m. fl. 2019).

Top-p Top-p, som också kallas *nucleus sampling*, är en samplingteknik framtagen av Holtzman m. fl. (2019) som även den påverkar vilken token som väljs härnäst i textgenerering. Till skillnad från top-k tittar man inte på ett fast antal toppkandidater, utan istället på den sammanlagda sannolikheten. Modellen beräknar sannolikheten $P(x_i)$ för varje möjlig nästa token x_i. Dessa sorteras sedan från högst till lägst sannolikhet, och summeras sedan tills tröskeln p nås. Antag att modellen har beräknat följande sannolikheter: token A har sannolikheten $0{,}4$, B har $0{,}3$, C har $0{,}25$, D har $0{,}025$ och E har $0{,}025$. Om tröskelvärdet p sätts till $0{,}9$, identifieras kärnan genom att summera sannolikheterna i fallande ordning tills summan överstiger detta värde. I detta fall ger summan $0{,}4 + 0{,}3 + 0{,}25 = 0{,}95$, vilket innebär att tokens A, B och C ingår i kärnan. Tokens D och E ignoreras och deras sannolikheter sätts till noll (Hyrkas 2023). Rent matematiskt uttrycks detta som att givet en fördelning $P(x|x_{1:i-1})$, så definieras dess top-p vokabulär $V^{(p)} \subset V$ som den minsta mängden så att:

$$\sum_{x \in V^{(p)}} P(x|x_{1:i-1}) \geq p \tag{9}$$

Alltså är skillnaden mellan top-k sampling och top-p sampling att top-k begränsar antalet

kandidater, medan top-p begränsar sannolikhetsmassan.

2.1.5 RAG

Stora språkmodeller, som kan bestå av flera miljarder parametrar, har en stor styrka i att kunskapsbasen är parametriserad och därmed inte behöver åtkomst till något externt minne eller ytterligare kunskap. Detta medför dock vissa nackdelar, såsom svårigheter att uppdatera eller utöka modellens kunskap, begränsad förmåga att ge tydliga förklaringar till sitt resonemang, samt risken för att generera felaktig eller påhittad information – så kallade ”hallucinationer”. Retrieval-Augmented Generation (RAG) är ett ramverk som kombinerar två komponenter - en retriever-del som hämtar de mest relevanta textdelarna från ett stort dokument- eller vektorindex, samt en ’vanlig’ språkmodell som formulerar ett svar och sätter det i rätt kontext utifrån den externa kunskapen den precis hämtat (P. Lewis m. fl. 2020). Genom att låta modellen läsa in kunskap i realtid i stället för att förlita sig enbart på vad som finns i parametrarna får man kunskap som går att uppdatera, spårbarhet via citerade källor och färre hallucinationer, samtidigt som man behåller den delen av språkmodeller som står för själva textgenereringen.

2.2 Promptingtekniker

Det finns många olika sätt att formulera och jobba med prompts till stora språkmodeller, och det går att argumentera för att detta har utvecklats till en egen vetenskap. Nedan följer en beskrivning av *prompt chaining*, *stepwise prompting*, och *meta prompting*. Dessa promptingtekniker har varit centrala för detta arbete.

2.2.1 Prompt chaining & Stepwise prompting

Prompt chaining är en teknik inom prompting som innebär att bryta ner ett relativt stort prompt till flera mindre prompts. Det första steget är att bryta ner ett problem till delproblem. Språkmodellen promptas sedan med att lösa det första delproblemet, varpå dess output blir input i språkmodellens nästa körning då den löser nästa delproblem. En tydlig fördel med att dela upp prompts i mindre beståndsdelar och låta en språkmodell generera en output för varje delmoment är att det ger högre transparens och kontroll. Att felsöka vad som inte gett önskat resultat i händelsekedjan blir betydligt lättare då det är möjligt att ha insikt i varje steg i processen (Prompt Engineering Guide 2025).

I en studie av Sun m. fl. (2024) undersöker författarna hur prompt chaining presterar i jämförelse med så kallad *Stepwise Prompting*. Stepwise prompting är en metod som står i kontrast till prompt chaining, där språkmodellen instrueras att utföra flera steg i en sekvens inom en enda generering. Studien gick ut på att dessa två metoder skulle användas för att sammanfatta textmaterial. Uppgiften som språkmodellen skulle hantera var att

1. Generera en initial sammanfattning

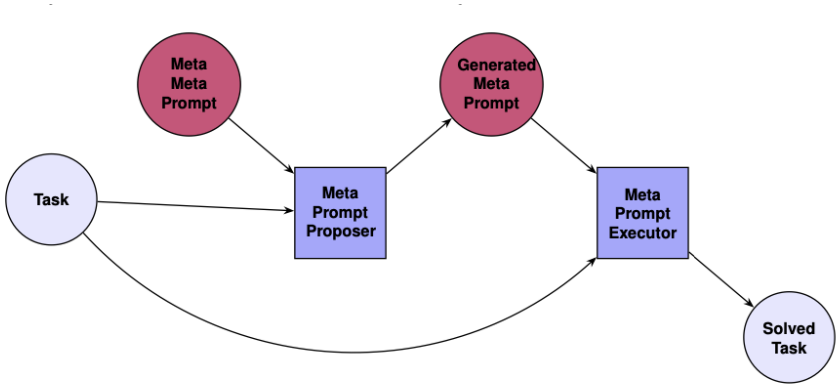
2. Ge kritik på den initiala sammanfattningen
3. Använda kritiken för att förbättra sammanfattningen

Först lät man lösa uppgiften med prompt chaining då de tre delstegen löstes separat med nya anrop till språkmodellen. Sedan löstes samma uppgift med stepwise prompting, där man kombinerade alla tre delar i ett enda prompt. För att sedan jämföra resultaten användes LLMCompare - ett språkmodellsbaserat verktyg som jämförde och utvärderade sammanfattningarna. Sammanfattningarna utvärderades på övergripande kvalitet, utelämnande av viktig information, samt om de innehöll överflödig irrelevant information. Studiens resultat visade att prompt chaining presterade bättre än stepwise prompting i 77 av 100 fall. Slutsatsen drogs att uppdelning av komplexa uppgifter i steg med separata prompts är mer effektivt än att hantera alla steg i ett enda prompt. Forskarna bakom artikeln slog även fast att resultaten sannolikt är överförbara till andra uppgifter inom textgenerering utöver att sammanfatta (Sun m. fl. 2024).

2.2.2 Meta prompting

Suzgun och Kalai (2024) presenterar i sin artikel ett tillvägagångssätt för prompting som de har valt att kalla för 'meta-prompting'. Den huvudsakliga tanken med meta-prompting är att nyttja en språkmodell för att koordinera och utföra flera oberoende förfrågningar till så kallade 'experter'. Processen går så till att en komplex uppgift ges som input till språkmodellen, som i sin tur agerar som en slags dirigent som bryter ner uppgiften i mindre deluppgifter, som utförs av experterna. Det är språkmodellens jobb att avgöra vilka experter som behöver konstrueras för att lösa den ursprungliga komplexa uppgiften som användaren har promptat modellen med (Suzgun och Kalai 2024). Varje expert är i sin tur byggd på samma språkmodell som dirigenten, fast med specifika instruktioner för att lösa just sin deluppgift. I artikeln specificeras inte huruvida detta ska ske i ett och samma anrop, men den genomgående idén är att experterna 'rapporterar tillbaka' till dirigenten.

Recursive Meta Prompting Zhang, Yuan och A. C.-C. Yao (2025) bygger vidare på teorin om meta prompting och talar om något de kallar 'Recursive Meta Prompting'. Författarna menar att de under de senaste åren har sett tillvägagångssätt som 'Chain-of-Thought'- och 'Tree-of-Thought'-prompting för att guida språkmodeller att artikulera avancerade resonemang steg för steg, men att dessa metoder inte fullt ut fångar den bredd och flexibilitet i den mänskliga kognitiva processen för att maximera prestandan när det gäller prompting. Chain-of-Thought-prompting innebär att modellen instrueras att redovisa sina resonemangssteg istället för att endast ge ett direkt svar. Detta har visat sig öka träffsäkerheten för modeller med över 100 miljarder parametrar inom flertalet problemlösningsorienterade typer av prompts (Wei m. fl. 2022). Tree-of-thought-prompting är en generalisering av Chain-of-Thought som innebär att problemlösningen beskrivs som ett sökträd där varje nod utgör en sammanhängande "tanke", så att språkmodellen kan utforska flera resonemangsvägar parallellt och med hjälp av egen heuristisk poängsättning bestämma när den ska gå vidare eller backa (S. Yao m. fl. 2023).



Figur 1: Recursive Meta Prompting (Zhang, Yuan och A. C.-C. Yao 2025)

Recursive Meta Prompting, RMP, handlar i grund och botten om att tillämpa meta-prompting på själva promptingprocessen (se figur 1). Med andra ord innebär det att språkmodellen inte bara löser en uppgift med hjälp av meta-prompting, utan även kan generera, förfina och förbättra sina egna prompts rekursivt. Det sker via en iterationsprocess där modellen använder sina egna outputs som inputs för att generera bättre prompts.

Zhao och Ma (2024) har i sin artikel identifierat styrkor och svagheter hos två olika designmetoder för prompts. Den ena är *LLM-Derived Prompts* (LDP), som delegerar prompt-designen helt till språkmodellen själv. Det finns där en risk att ett felaktigt steg i början av processen kan bilda en slags snöbollseffekt och leda till flera fel ju längre in i bearbetningen av det specifika problemet man kommer. Den andra metoden är *Expert-Designed Prompts* (EDP) vilket, som namnet antyder, handlar om promptstrukturer som är manuellt utformade av mänskliga experter. Medan det kan finnas fördelar med EDP, såsom att en människa naturligt har nära till strukturerade resonemangsspår samt en större möjlighet att validera och korrigera felaktiga delar av ett resonemang, kan det även innebära nackdelar. Dessa inkluderar att promptdesignern måste ha en djup förståelse av både själva problemdomänen och funktionaliteten som finns i språkmodeller. På så sätt blir prestationen begränsad av den mänskliga expertens kunskapsnivå (Zhao och Ma 2024).

Som en lösning på de nackdelar som finns med LDP och EDP har Zhao och Ma (2024) tagit fram ett ramverk de kallar *Prompt Recursive Search* (PRS). PRS fungerar på så sätt att språkmodellen utvärderar problemets komplexitet på en numerisk skala. Baserat på komplexiteten bryts problemet ner i lämpligt antal steg, och för varje mindre steg kan modellen tillämpa samma process om komplexiteten fortfarande är för hög.

2.3 Kunskapsgrafer

En kunskapsgraf är en datastruktur bestående av noder och relationer mellan dessa noder. Skillnaden från andra grafer är att relationerna mellan noderna har en semantisk betydelse och representerar verkligheten i någon mening, mer än att relationen existerar binärt.

2.3.1 Definition

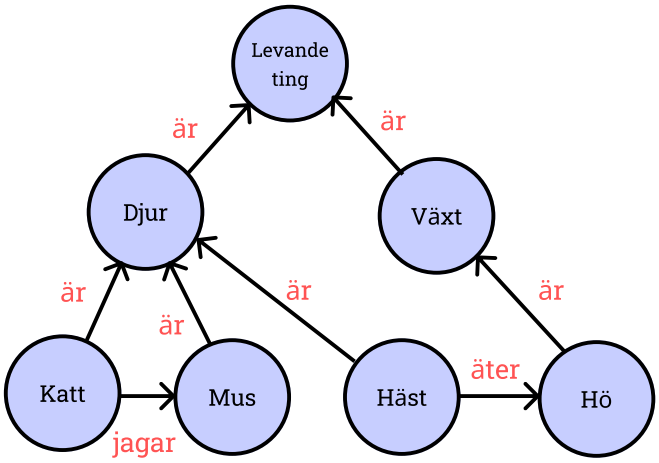
Exakt hur en kunskapsgraf definieras varierar, med vissa definitioner som rör sig åt mer specifika tekniska håll medan vissa andra är mer generella och inkluderande. Definitionen som Hogan m. fl. (2021) använder i sin gedigna artikel om kunskapsgrafer är följande:

A knowledge graph is a graph of data intended to accumulate and convey knowledge of the real world, whose nodes represent entities of interest and whose edges represent relations between these entities.

Ovanstående definition innebär alltså att en kunskapsgraf är en datastruktur som består av noder och relationer mellan dessa noder, och att dessa noder och relationer ska representera någon slags kunskap hämtad från den verkliga världen. Liknande definitioner förekommer även i annan litteratur. Ehrlinger och Wöß (2016) låter sin definition vara:

A knowledge graph acquires and integrates information into an ontology and applies a reasoner to derive new knowledge

Den senare definitionen strävar efter att särskilja definitionen av en kunskapsgraf från begreppet *knowledge base* (kunskapsbas). Författarna vill belysa det faktum att en kunskapsgraf inte bara kan visa *att* två saker har en relation, utan även *hur* och *varför* dessa är relaterade.



Figur 2: Enkel visuell representation av konceptet kunskapsgraf

Figur 2 visar ett enkelt exempel på hur noder och relationer i en kunskapsgraf skulle kunna se ut. Kunskapsgrafer lagras oftast i grafdatabaser, som internt lagrar relationerna mellan noderna. Noderna kan representera objekt, händelser, situationer, koncept, med mera och relationerna mellan dessa fångar kontexten i vilken de har med varandra att göra (Stegeman 2024).

2.3.2 Ontologi

Ett begrepp som nämns i definitionen av Ehrlinger och Wöß (2016) är *ontologi*. Ontologi defineras av Gruber (1995) som följande:

An ontology is an explicit specification of a conceptualization.

I ett försök att tolka denna definition, noterar Granström (2003) att ordet ontologi ursprungligen kommer från filosofin, men har med tiden kommit att användas inom bland annat artificiell intelligens och datastrukturer, och det är inom den senare nämnda domän som definitionen har tagits fram. Vidare menar hon att ontologin bland annat kan ses som en specifikation av en viss konceptualisering (Granström 2003). Genom att ha en ontologi, kan en enad begreppsapparat säkerställas så att alla system och personer som använder denna kan referera till samma entiteter och relationer på ett enhetligt sätt. En välkonstruerad ontologi kan ses som ett 'schema' för kunskapsgrafen, så att alla noder och relationer följer de definitioner som är specificerade i ontologin (Noy, McGuinness m. fl. 2001).

2.3.3 Tillämpningar

Kunskapsgrafer har ett flertalet praktiska tillämpningar inom olika domäner, bland annat inom sökmotorer, bioinformatik, företagssystem, och rekommendationssystem. Ett exempel på just ett rekommendationssystem är e-handelsjätten Amazon som använder sig av kunskapsgrafer. Företaget har som mål att leverera produktrekommendationer som är så relevanta som möjligt för kundens sökningar, vilket ibland kräver sunt förnuft. Om en kund exempelvis söker efter 'skor för gravida kvinnor', ska rekommendationssystemet kunna avkoda att det som menas är att gravida kvinnor kanske vill ha skor med halkskydd. För att kunna göra dessa implicita kopplingar mellan sökningar och önskade produkter använder sig Amazon av kunskapsgrafer, så att produkter i deras katalog kan länkas samman med de mänskliga kontexter som de existerar i. I detta exempel hade en relation av typen 'används för målgrupp' kunnat länka samman produkten 'skor med halkskydd' och kundsegmentet 'gravid kvinna' (Yu och Li 2024).

3 Metod

Arbetet genomfördes i tre faser. De första två faserna ämnade att besvara den första forskningsfrågan, gällande huruvida en fullt automatiserad process med hjälp av stora språkmodeller kan producera meningsfulla kunskapsgrafer. Den tredje fasen ämnade att besvara den andra forskningsfrågan, som gällde huruvida olika parameterinställningar och varianter av prompts kan ha en signifikant påverkan på processen. De tre faserna beskrivs överskådligt nedan:

1. I arbetets första fas undersöktes möjligheterna att med hjälp av metoden *stepwise prompting*, skapa en kunskapsgraf för en hel universitetskurs, i ett enda anrop till språkmodellen. Stepwise prompting går ut på att lösa ett problem genom att dela upp det i mindre delproblem - men fortfarande ge instruktionerna för dessa delproblem i ett och samma prompt.

2. I den andra fasen undersöktes möjligheten att skapa en kunskapsgraf för en hel kurs med metoden *prompt chaining*. Prompt chaining skiljer sig från stepwise prompting i att man istället för att behandla hela problemet i ett prompt, länkar samman flera anrop till språkmodellen där utdata från ett anrop fungerar som indata till nästa anrop. Varje anrop behandlar en definierad deluppgift.

I studien som redovisas i kapitel 2.2 dras slutsatsen att prompt chaining presterar bättre vad gäller uppgifter relaterade till textbehandling än vad stepwise prompting gör. Studien i fråga hade dock inget att göra med kunskapsgrafer, och det fanns därför anledning att undersöka hur stepwise prompting presterar i denna kontext. Båda metoderna kommer med sina inneboende för- och nackdelar. Med stepwise prompting begränsas antalet anrop vilket kan ha en kostnadsmässig fördel, men ger sämre insyn i språkmodellens process. Prompt chaining ger bättre insyn och enklare felsökning, men det sker dock på bekostnad av ett ökat antal anrop till språkmodellen.

För att utvärdera graferna som skapades med dessa två metoder gjordes ett val att analysera dem *kvalitativt*. Det *är* möjligt att jämföra grafer i kvantitativa mått, med metoder som exempelvis *belief propagation* (Koutra m. fl. 2011). Däremot ansågs det inte lämpligt att göra det i denna studie av anledningen att det som främst skulle ge värde av en sådan analys är att jämföra likheten mellan de genererade graferna och en 'facit'-graf. Eftersom det låg utanför denna studies omfattning att konstruera en 'facit'-graf, då det kräver specifik matematisk expertis, sågs det som ett rimligare tillvägagångssätt att kvalitativt analysera specifika exempel som belyser observerade fenomen i de skapade graferna.

Arbetets tredje fas bestod av att närmare undersöka ett av stegen i prompt chaining-flödet, närmare bestämt det steget som identifierar nyckelbegrepp i texterna:

3. En testingfas som för flertalet matematiska teorikaptiel, inte ämnade att skapa en hel kunskapsgraf utan endast genomföra den första deluppgiften - att extrahera nyckelbegrepp. Här gjordes en analys på vilken påverkan på stabiliteten i outputen det har att använda olika mängd detaljerade instruktioner i promptet, samt variera värden på de olika hyperparametrarna.

3.1 Fas 1 - Stepwise prompting

För att utforma promptet som skulle generera kunskapsgrafen användes metoden *recursive meta prompting* som beskrivs i kapitel 2.2.2. En initial problembeskrivning gavs till språkmodellen, som sedan utformade ett **system prompt** till språkmodellen. Ett system prompt är inom stora språkmodeller en initial fördefinierad instruktion eller kontext som ges till modellen för att styra dess beteende under interaktionen och hur den ska leverera utdatan (von Reybekiel och Wennström 2024). I systempromptet kan det även vara fördelaktigt att formulera ett exempel, så att modellen bättre får en förståelse för vilken utdata användaren vill ha. Detta kallas i formella termer för **in-context learning** (Wies, Levine och Shashua 2023). Systempromptet är det som placeras först i anropet till språkmodellen och är alltså särskilt från det som kallas för **user prompt**, den instruktion som användaren matar in. För att konkretisera detta med ett exempel skulle följande mål kunna formuleras: skapa en AI-tjänst som förklarar svårbegriplig matematik för barn. Systempromptet skulle då kunna lyda:

> Du är en pedagogisk expert som ska hjälpa barn i mellanstadieåldern med matematiska uppgifter. Tänk på att barn inte har en bra förståelse för den matematiska domänen, och du måste därför anpassa dina förklaringar så att den givna målgruppen förstår lösningen.
>
> Exempel: Om användaren skriver:
>
> ’Jag förstår inte ekvationen 5x = 100’,
>
> skulle ditt kunna svar vara:
>
> ”Jag förstår att det där med bokstäver i matte kan vara lite knepigt! X betyder att vi inte vet vilken siffra som ska stå där. Vi vet bara att 5 gånger någonting (x) ska bli 100. Tänk dig att du och dina 4 kompisar köper lördagsgodis för totalt 100 kronor. För att veta hur mycket varje person ska betala måste vi dela 100 med 5 - och vi får då reda på x!”

För att generera prompten användes språkmodellen Claude, som beskrivs i 3.5.1. Konversationen med språkmodellen inleddes med en enkel förfrågan, som kan ses nedan i exempel 1.

```
Du ska hjälpa mig att skriva ett prompt till en
LLM likt Claude.
Målet är att jag ska mata in en kort text som
kärnbegrepp ska extraheras ifrån. Om jag exempelvis
matar in texten:
[EXEMPELKAPITEL]
vill jag få ut kärnbegreppen:
[LISTA MED KÄRNBEGREPP]
```

Exempel 1: Den första instruktionen som gavs till Claude för att börja generera ett prompt

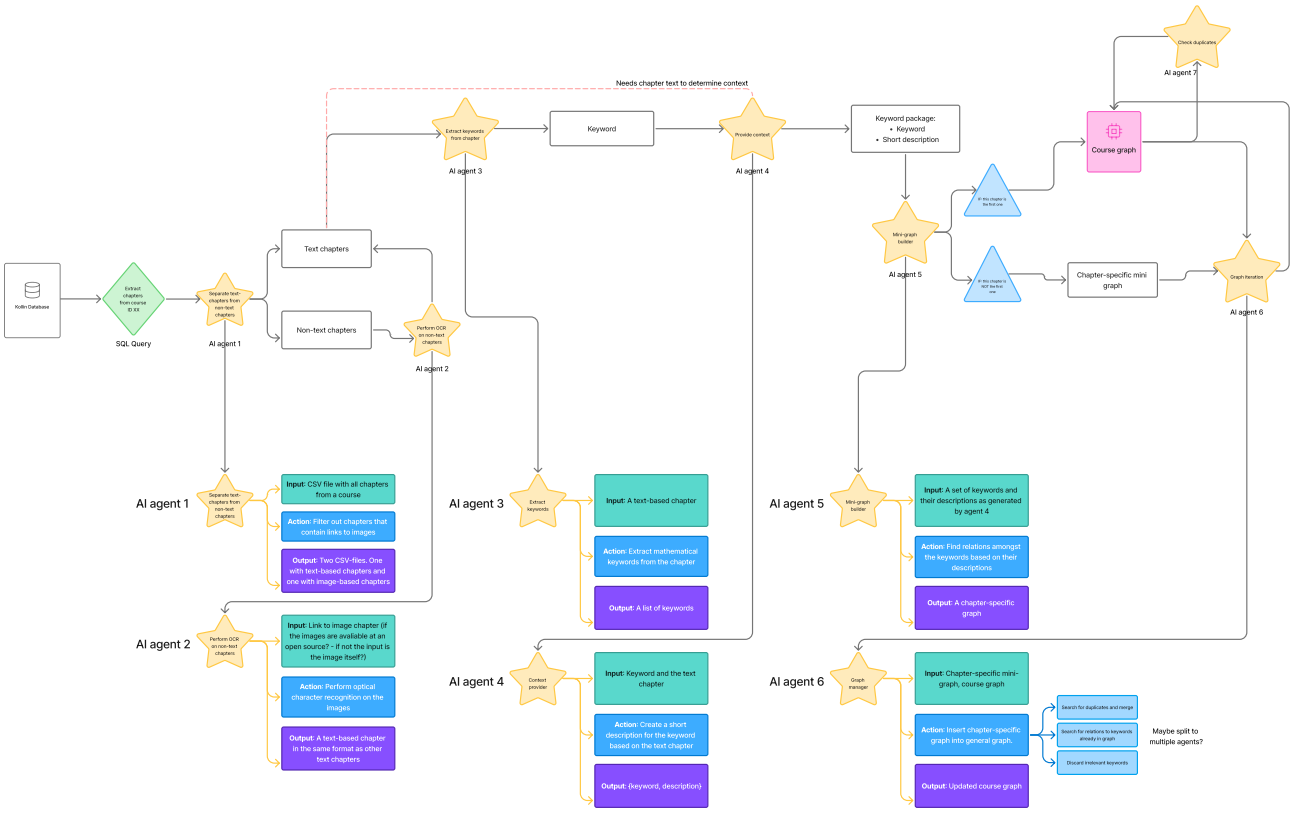
Utifrån denna första förfrågan skapade språkmodellen en första version av promptet som kan ses i 4.1. När språkmodellen skapat detta system prompt, som till en början skulle kunna användas för att extrahera nyckelord från en text, påbörjades en iterativ process för att generera ett mer omfattande prompt som skulle kunna generera all data för hela kunskapsgrafen i en enda körning. För att åstadkomma detta användes Recursive Meta Prompting, där språkmodellen med hjälp av nya direktiv och förslag på justeringar förändrade sitt ursprungsprompt för att uppfylla nya krav i skapandet av grafen. De inputs som gavs till språkmodellen under den iterativa processen var i stora drag följande:

- Baserat på de inmatade matematiska texterna, identifiera relationer mellan begreppen
- Relationerna ska vara riktade och gå från mer grundläggande begrepp till mer avancerade
- Se till att det inte finns några dubbletter och slå ihop begrepp som är grammatiska variationer av varandra
- Se till att inga par av begrepp har mer än en relation mellan varandra
- Att i enlighet med ’In-context learning’ (se 3.1) inkludera exempel i promptet.

Under processen matades en CSV-fil med textkapitel samlade från en kurs in som input till språkmodellen för att iterativt utvärdera hur varianterna av promptet presterade. Efter granskning av de genererade graferna identifierades ytterligare behov som behövde adresseras, varpå Recursive Meta Prompting fortsatte att tillämpas för att få med de krav och behov som eftersträvades. Exempel på mindre iterationer, utöver de mer övergripande som anges i listan ovan, kunde bland annat inkludera att säkerställa att alla begrepp hade minst en relation till ett annat begrepp, reglera vilka typer av relationer som skulle genereras, extra valideringssteg, och kartlägga hur många relationer varje begrepp har. Totalt skedde **62** iterationer av språkmodellen för att nå det ’slutgiltiga’ promptet. Notera att flertalet av dessa iterationer gjorde språkmodellen i en sekvens utan mänsklig input emellan. Till exempel, om mänsklig input gavs för att justera någon del av promptet, kunde modellen iterera promptet mer än en gång innan en version gavs som output.

3.2 Fas 2 - Prompt chaining

Vid användning av prompt chaining är alltså tanken att dela upp ett huvudproblem i mindre problem som behandlas av flera prompts istället för ett prompt. För att modellera hur denna kedja skulle se ut gjordes en grov skiss av flödet, som börjar med att hämta teorikapitlen i Kollins databas, ända fram till en färdig kunskapsgraf. Skissen utformades delvis med hjälp av meta prompting, alltså att språkmodellen gav assistans i att bryta ner problemet i mindre delar och ge förslag på vilka beståndsdelar som bör ingå i modellen. I skissen används begreppet 'AI agents'. En AI-agent är i grund och botten ett autonomt system som självständigt utför uppgifter. De bygger ofta på stora språkmodeller och kan användas för att effektivisera många olika typer av processer (Gutowska u. å.). I denna kontext kommer de enskilda blocken som hanterar varje delprompt benämnas som AI-agenter. Skissen ses nedan i figur 3, och i full storlek i bilaga A.



Figur 3: Första skiss av Prompt Chaining-systemet

Flödet i figur 3 börjar med att teorikapitlen extraheras från Kollins databas, varpå de behandlas i olika steg av sju stycken AI-agenter. I figuren är agenterna markerade som stjärnor. Självaste kunskapsgrafen är den rosa rutan i det högra övre hörnet. Figuren tjänar syftet att ge en överblick över processen, texterna i figuren motsvarar det som nedan beskrivs. De huvudsakliga uppgifterna som de sju agenterna skulle utföra var följande:

- Agent 1 - separera textkapitel från kapitel som inte var textbaserade - detta kunde exempelvis handla om kapitel i bildformat.
  - Input till denna agent är en CSV-fil som innehåller alla kapitel (eller ett utvald antal) från en specifik kurs.
  - Output är två CSV-filer, en med textbaserade kapitel och en med kapitel som hänvisar till bilder.
- Agent 2 - utföra Optical Character Recognition (OCR) på de kapitel i bildformat för att kunna konvertera dem till samma format som kapitlen på textform
  - Input till denna agent är bilder eller bildlänkar till kapitel som inte är textbaserade
  - Output är kapitel på samma format som de textbaserade kapitlen
- Agent 3 - extrahera viktiga nyckelord från ett kapitel
  - Input till denna agent är ett textbaserat kapitel
  - Output är en lista med nyckelord i JSON-format
- Agent 4 - ge nyckelorden en kort beskrivning baserat på kapitlet de extraherats från
  - Input till denna agent är textkapitlet som även gavs som input i agent 3, samt listan med nyckelord som gavs som output från samma agent
  - Output är en lista med objekt, där varje objekt har attributen ’nyckelord’, och ’beskrivning’.
- Agent 5 - bygg en ’mini-graf’ för det specifika kapitlet genom att ge relationer mellan begreppen baserad på beskrivningarna som genererats av agent 4
  - Input till denna agent är listan med nyckelorden och dess tillhörande beskrivningar som gavs som output från agent 5
  - Output är en lista med objekt som innehåller relationerna. Varje objekt har attributen ’start’, ’end’ och ’relationstyp’
- Agent 6 - integrera mini-grafen i kurs-grafen som innehåller de tidigare identifierade begreppen och relationerna
  - Input till denna agent är den kapitelspecifika output från agent 5, samt kursens ’stora’ kunskapsgraf
  - Output är en uppdaterad version av kursens kunskapsgraf
- Agent 7 - ett sista kontrollsteg som validerar grafen genom att hitta och rensa dubbletter och andra felaktigheter.
  - Input till denna agent är hela kursens kunskapsgraf
  - Output är en validerad och uppdaterad version av grafen

Likt när promptet för processen med Stepwise prompting skapades användes Recursive Meta Prompting för att generera prompts till de individuella agenterna. Under processen togs beslutet att exkludera stegen som beskrevs i agent 1 och agent 2 ovan. Anledningarna till detta var dels att teorikapitlen i Kollins databas som är bildbaserade istället för textbaserade är i minoritet och Kollin har inte för avsikt att i framtiden addera fler teorikapitel i detta format. En annan anledning är att arbetet hade hamnat något utanför det önskade fokuset att bygga grafen med hjälp av språkmodeller och inte andra AI-baserade verktyg. Precis som med stepwise prompting så redovisas de genererade prompten i resultatet under kapitel 4.2.

3.3 Fas 3 - Nyckelordsextrahering

I detta steg var syftet att göra en undersökning av till vilken grad olika promptvarianter och hyperparameterinställningar har på outputen. Ett relevant begrepp att ta hänsyn till här är 'Garbage in - Garbage out' (GIGO). Det är en grundläggande princip inom dataanalys som i grund och botten handlar om att kvaliteten på utdatan är beroende av kvaliteten på indatan. Om 'garbage', det vill säga data med låg kvalitet, används som input i ett system, kommer systemet med högsta sannolikhet även att ge lågkvalitativ output. Vad som menas med just lågkvalitativ data kan innebära flera olika saker, bland annat felaktig data, data som är för avvikande från övrig data (även benämnt som 'outliers'), data som är för lik annan data, saknad data, eller data som inte är relevant eller applicerbar i den givna kontexten (Awati 2023). Motiveringen att just göra en djupdykning i nyckelordsextraheringen är att kvaliteten på den färdiga grafen, enligt 'GIGO'-principen, inte kommer vara högre än kvaliteten på utdatan från det första steget i processen. För att applicera de olika kategorierna av 'garbage data' i steget med nyckelordsextrahering skulle det i detta fall kunna handla om att irrelevanta nyckelord har extraherats, relevanta nyckelord saknas, det förekommer nyckelord som är dubbletter alternativt är grammatiska varianter av varandra. För att optimera steget som just behandlar nyckelordsextraktion sattes en testprocess upp, som bestod av följande steg:

1. 3 varianter av systempromptet som instruerade språkmodellen att extrahera nyckelord formulerades (för att se alla varianter i sin helhet - se bilaga E):
    - En 'standard'-variant som var en kopia av det prompt som hittills använts i detta steg
    - En 'minimal'-variant där instruktionerna i promptet skalats av
    - En 'detailed'-variant där instruktionerna i promptet var fler och tydligare
2. En uppsättning av hyperparameterkombinationer sattes samman
3. För varje kombination av promptvariant och hyperparametrar, anropades språkmodellen 7 gånger. Likheten mellan dessa 7 körningar analyserades sedan.
4. Samma procedur genomfördes för flera teorikapitel för att kunna göra en generell analys på hur olika promptvarianter och hyperparameterinställningar påverkar stabiliteten i nyckelordsextraheringen.

Hyperparametrarna som användes redovisas nedan i tabell 2

.

Tabell 2: Hyperparameterkombinationer för tester av nyckelordsextrahering

Id	Temperatur	Top-k	Top-p
1	0	10	0,5
2	0	40	0,9
3	0	100	1
4	0,3	20	0,8
5	0,3	40	0,9
6	0,3	100	1
7	0,5	20	0,8
8	0,5	40	0,9
9	0,5	100	1
10	0,7	20	0,8
11	0,7	40	0,9
12	0,7	100	1
13	1	20	0,8
14	1	40	0,9
15	1	100	1

Testerna gjordes på totalt **21 teorikapitel**, från olika matematiska kurser. Detta resulterade i 21·3·15·7 = **6 615** anrop till språkmodellen (3 promptvarianter, 15 hyperparameterkombinationer, 7 körningar för varje grupp). Detta gjordes genom att anropa språkmodellen via dess API i en Python-loop. Pseudokoden för denna loop kan ses i bilaga B. Analysen på resultatet från denna fas gjordes både kvantitativt och kvalitativt. Den kvantitativa delen bestod av att analysera vilka värden på hyperparametrarna som förändrade likheten mellan körningarna signifikant, medan den kvalitativa delen bestod av en stickprovsanalys på de faktiska nyckelorden som extraherats.

3.4 Statistiska mått

I arbetet används ett antal statistiska mått, som gås igenom i detta avsnitt.

3.4.1 Jaccardlikhet

I steg 3 som beskrivs ovan analyserades likheten mellan körningarna med måttet jaccardlikhet, vilket är ett mått för att jämföra likheten mellan två objekt. Jaccardlikheten kan inom den matematiska domänen beräknas mellan två binära vektorer eller mellan två mängder, men kan likväl appliceras i dataanalys för att analysera likheten mellan två textdokument (Karabiber u. å.). Beräkningen av Jaccardlikheten ser ut som följer:

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|} \tag{10}$$

där $|A \cap B|$ anger hur många enheter som delas mellan båda mängderna A och B, medan $|A \cup B|$ anger det totala antalet enheter i båda mängderna. Om mängderna inte delar några enheter kommer Jaccardlikheten att vara 0, däremot om de är identiska kommer värdet att vara 1.

3.4.2 Spearmankorrelation

För att beräkna hyperparametrarnas inflytande på resultatet använde Spearmans korrelationskoefficient. Spearman föredras framför exempelvis Pearsons korrelationskoefficient när datan inte uppfyller antaganden om kontinuerliga, normalfördelade variabler, vilket inte hyperparameteruppsättningarna i tabell 2 gör (Schober, Boer och Schwarte 2018). Formeln för att beräkna Spearmans korrelationskoefficient r_s är:

$$r_s = 1 - \frac{6 \sum_{i=1}^{n} d_i^2}{n(n^2 - 1)} \tag{11}$$

där $d_i = R[X_i] - R[Y_i]$ är differensen mellan varje observationers rang på variablerna X och Y, och n är antalet observationer.

3.4.3 R^2

För att undersöka om det fanns något samband mellan antal extraherade nyckelord och likhetsindex användes regressionsanalys. För att utvärdera hur bra datan passar en regressionsmodell användes R^2 som mått. R^2-skalan går mellan -1 och 1, och ett R^2 värde nära 1 alternativt -1 indikerar generellt att modellen förklarar variationen i data bra, medan ett värde nära 0 visar det motsatta. Däremot kan olika måttstockar användas i olika kontexter. R^2 ¿ 0,8 anses vara högt i rent kvantitativa fält där mätfelen är små, medan det i sociala och beteendevetenskapliga studier kan anses vara högt med ett R^2-värde i spannet 0,3-0,6 (Nau u. å.). Formeln för att beräkna R^2 är:

$$R^2 = 1 - \frac{\sum_{i=1}^{n} (y_i - \hat{y}_i)^2}{\sum_{i=1}^{n} (y_i - \hat{y})^2} \tag{12}$$

där y_i är det observerade värdet, $\hat{y}_i$ är det predikterade värdet från regressionsmodellen, och $\hat{y}$ är medelvärdet av alla observerade värden y_i.

3.4.4 Kruskal-Wallis-test & Dunns Post-hoc-analys

För att undersöka huruvida temperaturnivån hade en signifikant påverkan på Jaccardlikheten inom respektive promptvariant användes Kruskal–Wallis-testet, som lämpar sig väl för situationer där antaganden om normalfördelning inte är uppfyllda, då det istället bygger på rangordningar snarare än absoluta värden (Dunn 1964). Analysen genomfördes

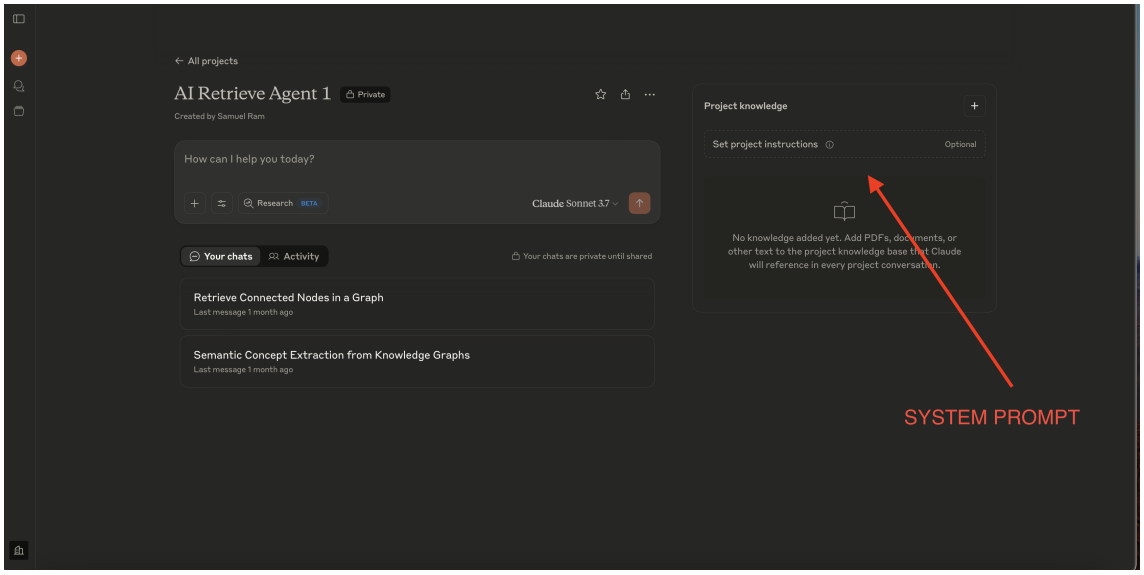
separat för varje promptvariant (minimal, standard, detaljerad), med temperatur som oberoende variabel och genomsnittlig Jaccardlikhet som beroende variabel. Kruskal–Wallis-testet testade om det fanns minst en temperaturgrupp som avvek signifikant från de övriga. När en signifikant skillnad identifierades ($p < 0.05$) genomfördes ett post-hoc-test med Dunn's test för parvisa jämförelser mellan temperaturer. Fokus låg på att identifiera den lägsta temperatur som avvek signifikant från t = 0.0, i syfte att etablera ett tröskelvärde där temperatur börjar ha en påvisbar effekt på Jaccard-likheten.

3.5 Verktyg

För att kunna genomföra arbetet användes huvudsakligen följande tre verktyg: den stora språkmodellen Claude, grafdatabsen Neo4j, och arbetsflödesverktyget n8n. De beskrivs i korthet nedan.

3.5.1 Claude

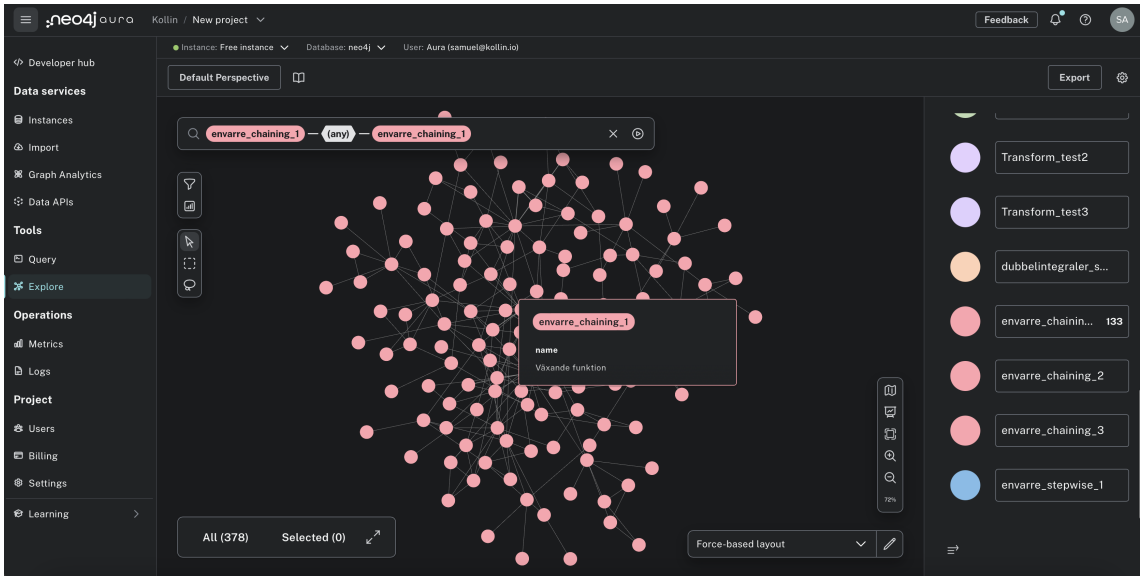
Den stora språkmodellen som använts i arbetet har varit Claude Sonnet 3.7 - en modell av företaget Anthropic. Claude går att använda i Anthropics webbgränssnitt, och företaget har också lanserat en API som gör att Claude kan användas av tredjepartsutvecklare. Anthropic har för nuvarande tre olika Claude-modeller; Claude Haiku, Claude Opus, och Claude Sonnet. Haiku beskrivs som Anthropics snabbaste och lättare modell, som lämpar sig för enklare uppgifter. Opus är den mest högpresterande modellen, som kan göra komplexa analyser och passar bäst för svåra matematiska och kodningsrelaterade uppgifter. Sonnet marknadsförs som en mellanväg av de två tidigare nämnda modellerna, med en kombination av hastighet och prestation (Anthropic 2025). Valet att använda Claude i detta projekt baserades huvudsakligen på att Kollin redan hade ett konto hos Anthropic, vilket underlättade implementationen via API. Figur 4 visar en skärmdump från Claudes webbapplikation. Den röda pilen i figuren anger var det går att ange 'project instructions', vilket är samma sak som system prompt som går att läsa om i 3.1.



Figur 4: Skärmdump av Claudes webbapplikation

3.5.2 Neo4j

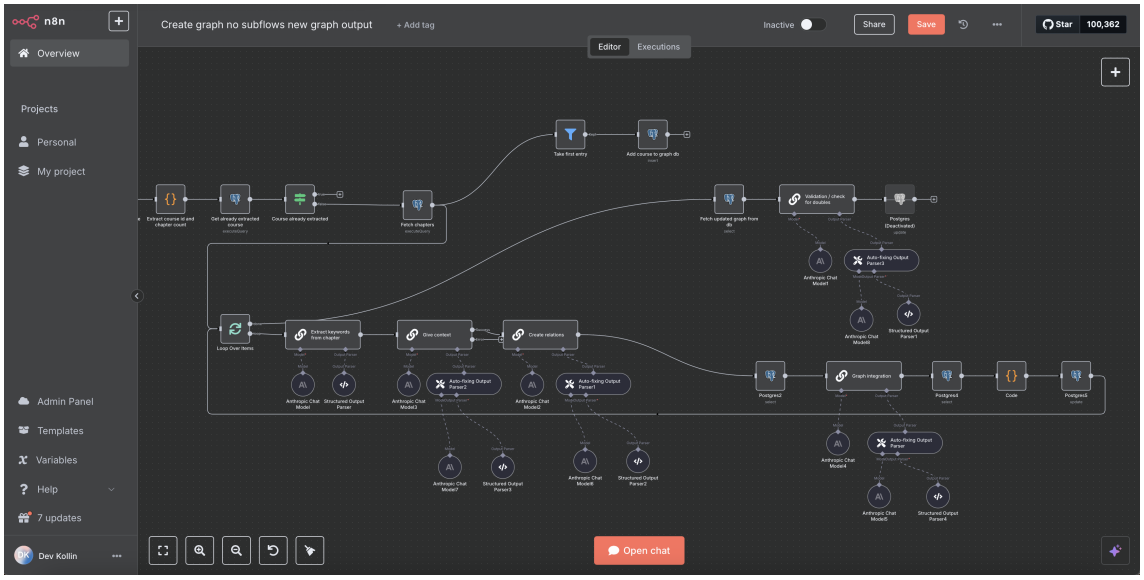
Neo4j är en grafdatabas som används för att lagra, hantera, och analysera information i form av grafer bestående av noder och länkar mellan dessa i form av relationer. Neo4j använder språket Cypher för att skapa, läsa, uppdatera och ta bort data i grafer på ett sätt som gör det enkelt att uttrycka komplexa frågeställningar, som t.ex. ”vilka personer känner någon som bor i samma stad som jag?”. Cypher gör det möjligt att fokusera på vad man vill hämta snarare än hur man tekniskt navigerar datan, vilket gör det enkelt att använda för både tekniska och icke-tekniska användare (Neo4j Inc. 2025). Inom ramen för detta arbete har Neo4j främst använts för att visualisera de grafer som skapats. I den form som graferna existerat i grafskapandeprocessen har det handlat om två textfiler med noder respektive relationer (se 3.1), och det var från dessa inte helt tydligt att utläsa grafens struktur och omfattning. Med hjälp av Neo4j kunde dessa textfiler i csv-format enkelt importeras till en grafdatabas och sedan visualiseras direkt i webbapplikationens gränssnitt (figur 5 visar en skärmdump av detta). Figurerna i kapitel 4.1.2 och 4.2.2 är genererade från Neo4j.



Figur 5: Skärmdump av Neo4js gränssnitt

3.5.3 n8n

n8n är ett verktyg för automatisering av arbetsflöden. Det har sina styrkor i sitt användarvänliga gränssnitt i en low-codemiljö samtidigt som möjligheter finns att använda anpassad kod vilket ger verktyget en högre grad av flexibilitet. n8n har över 500 integrationer med olika tjänster, vilket innebär att det går att koppla samman en mängd olika system, såsom Slack, Google Sheets, Github, och många fler (n8n 2025). I n8n är det dessutom möjligt att använda sig av AI-agenter, med lättintegrerade API-anrop till olika stora språkmodeller. n8n användes som verktyg när prompt chaining utforskades, där en AI-agent kunde byggas för varje prompt i kedjan. I n8n är det också möjligt att enkelt koppla på så kallade *output parsers* på anropen till språkmodellen, vars uppgift är att tvinga in utdatan i ett visst format. Detta var användbart då en AI-agents utdata fungerade som nästa agents indata i prompt chaining-flödet. Som exempel tvingade output parsern in utdatan från nyckelordsextraheringen in i ett listformat, vilket innebär att utdatan ’rensas’ från allt vardagligt språk som vanligtvis följer med ett svar från en stor språkmodell. På så sätt underlättade det att använda denna lista som indata i nästa anrop, och samma princip applicerades på alla följande agenter. Givet att n8n även har stöd för integrering av Neon, den databastjänst som Kollin använder sig av, var det möjligt att direkt i n8n-flödet hämta teorikapitlen från databasen, för att sedan skapa grafen i samma flöde, utan att behöva skriva anpassad kod. I figur 6 visas det flöde som tagits fram i tjänsten utefter skissen i figur 3.



Figur 6: Skärmdump av arbetsflödet i n8n

4 Resultat

Nedan redovisas resultaten från både stepwise prompting och prompt chaining. Detta inkluderar både de prompts som genererats för att skapa graferna, och själva graferna i sig. Efter det följer resultaten från testprocessen av nyckelordsextraheringen.

4.1 Stepwise prompting

Nedan redovisas först två versioner av det systemprompt som användes i stepwise prompting-processen, följt av de grafer som genererats med metoden.

4.1.1 Promptgenerering

Det första promptet som genererades utifrån den initiala förfrågan till Claude som beskrivs i 3.1 kan ses i exempel 2.

```
Matematisk begreppsextrahering
Du är en assistent specialiserad på att identifiera
och extrahera kärnbegrepp
från matematiska texter på universitetnivå.
Din uppgift:

Analysera den matematiska texten som tillhandahålls nedan
Identifiera och extrahera matematiska kärnbegrepp som
är centrala för att förstå och tillämpa materialet
Presentera dessa begrepp i CSV-format

Vad som utgör ett kärnbegrepp:

Centrala matematiska koncept som definieras eller förklaras
i texten. Metodologiska termer som beskriver hur man utför
statistiska eller matematiska tester
Viktiga variabler, hypoteser eller modeller som är
väsentliga för det matematiska området
Specialiserade termer som skulle behöva förstås av en
student som studerar ämnet

Vad som INTE utgör ett kärnbegrepp:

Grundläggande matematiska operationer
(t.ex. addition, multiplikation)
Allmänna ord utan specifik matematisk betydelse
Vanliga statistiska termer som inte är specifika
för den aktuella kontexten
Begrepp som nämns i förbigående men inte utvecklas eller
är centrala för texten

Output-format
Extrahera begreppen och formatera dem som en enkel lista
med ett begrepp per rad, redo för CSV-export.
Analysera följande text:
[INFOGA MATEMATISK TEXT HÄR]
```

Exempel 2: Det initiala promptet för att skapa grafen med stepwise prompting

Som beskrivits i 3.1 skedde sedan en iterativ process med recursive meta prompting för att generera ett prompt som uppfyllde nya krav och behov som identifierades under processens gång. Det slutgiltiga promptet går att se i bilaga C.

4.1.2 Kunskapsgrafer genererade med stepwise prompting

För att testa metoden i liten skala användes till en början endast ett kapitel för att generera grafen, för att sedan testas på hela kurser bestående av flera kapitel. Ett exempel går att se i figur 7 som visar en kunskapsgraf som är genererad baserat på följande kapitel om variabelsubstitution i dubbelintegraler från en kurs i flervariabelanalys, som kan ses nedan i exempel 3:

Sats: Antag att vi vill beräkna en dubbelintegral där vi övergår från ett område $\mathcal{D}$ i xy-planet till ett område Ω i uv-planet. Antag att variabelsubstitutionen sker enligt

$$\begin{cases} x = x(u,v) \\ y = y(u,v) \end{cases}$$

Då gäller

$$\iint_{\mathcal{D}} f(x,y)\,dx\,dy = \iint_{\Omega} f(u,v)\,\kappa\,du\,dv$$

där skalfaktorn κ ges av absolutbeloppet av en viss Jacobideterminant:

$$\kappa = \left|\det \frac{\partial(x,y)}{\partial(u,v)}\right| = \left|\det \begin{bmatrix} \frac{\partial x}{\partial u} & \frac{\partial x}{\partial v} \\ \frac{\partial y}{\partial u} & \frac{\partial y}{\partial v} \end{bmatrix}\right|$$

Anmärkning: Om vi byter från variablerna x och y till variablerna u och v, ska skalfaktorn κ beräknas enligt

$$\kappa = \left|\det \frac{\partial(x,y)}{\partial(u,v)}\right|$$

dvs.

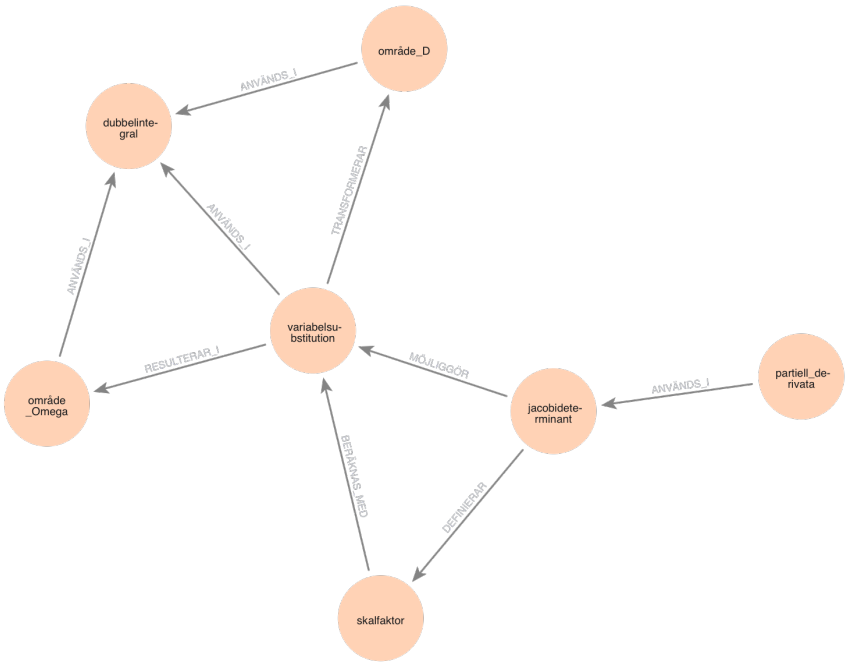
$$\kappa \neq \left|\det \frac{\partial(u,v)}{\partial(x,y)}\right|$$

hur lockande det än är.
Ett otroligt användbart samband mellan dessa två determinanter som kan komma till stor hjälp i många uppgifter är

$$\det \frac{\partial(x,y)}{\partial(u,v)} = \frac{1}{\det \frac{\partial(u,v)}{\partial(x,y)}}$$

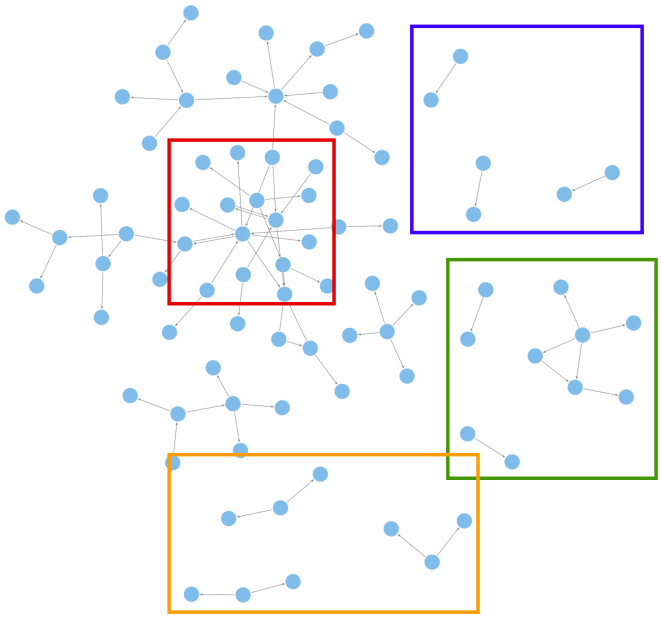
förutsatt att ingen av determinanterna är lika med 0.

Exempel 3: Teorikapitel från Kollins databas om variabelsubstitution, hämtat från en kurs i flervariabelanalys

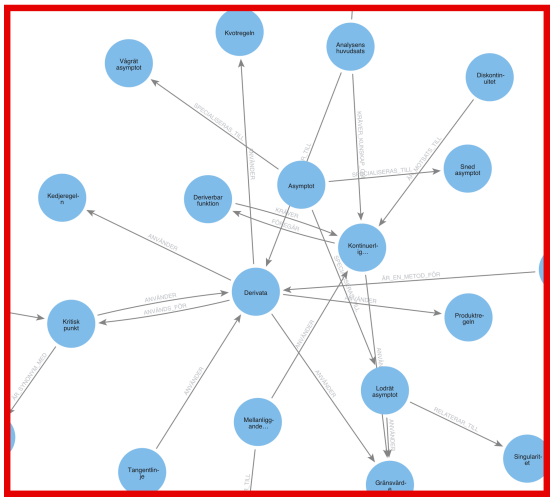


Figur 7: Visuell representation av kunskapsgraf genererat på ett inputkapitel om variabelsubstitution i dubbelintegraler.

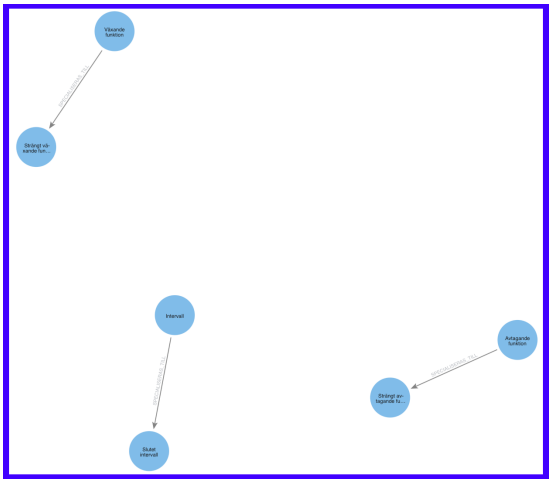
Ett exempel på en kunskapsgraf som genererades på en hel kurs visas i figur 8, där en kurs i envariabelanalys användes. De färgade rutorna förklaras i figur 9a - 9d.



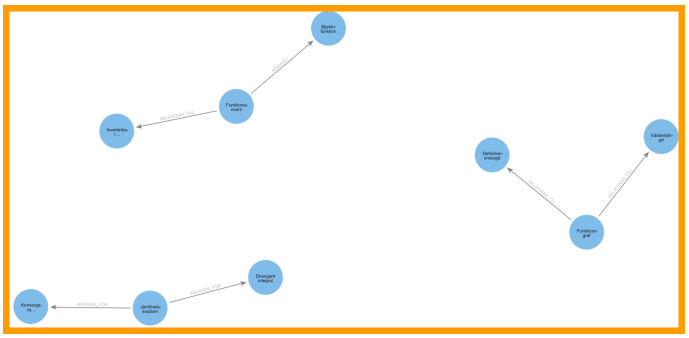
Figur 8: Visuell representation av kunskapsgraf genererat på alla kapitel från en kurs i envariabelanalys.



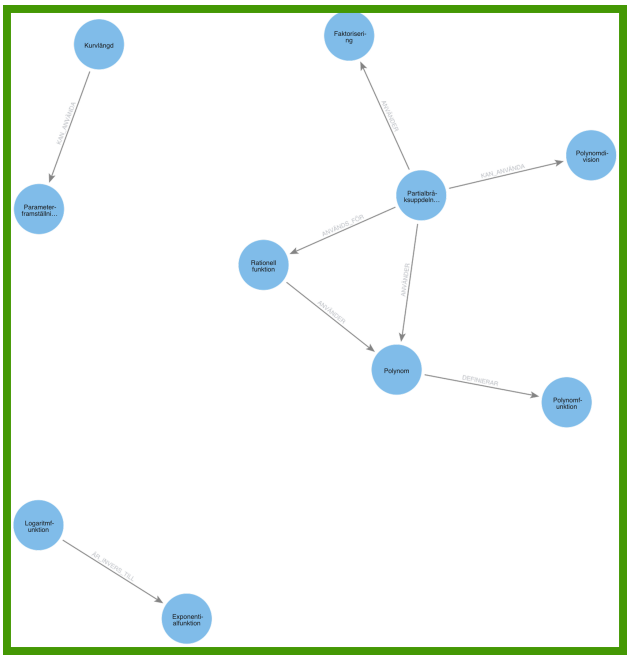
(a) Kluster som visar flertalet dubbelriktade relationer



(b) Tre stycken isolerade ’öar’ som endast består av två begrepp vardera



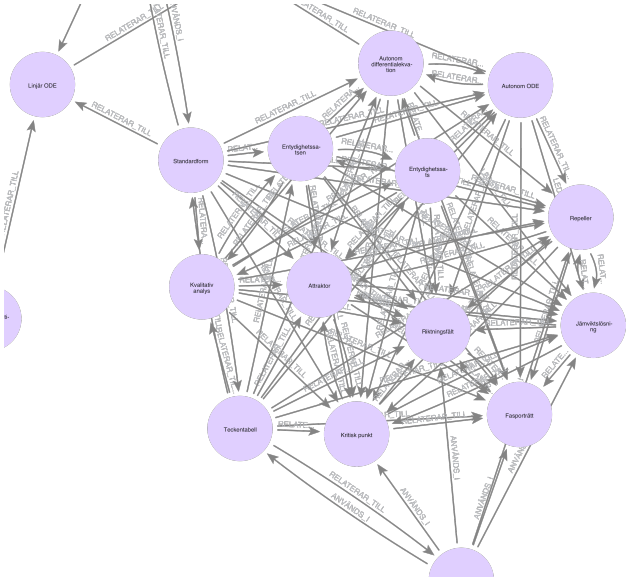
(c) Ytterligare isolerade öar



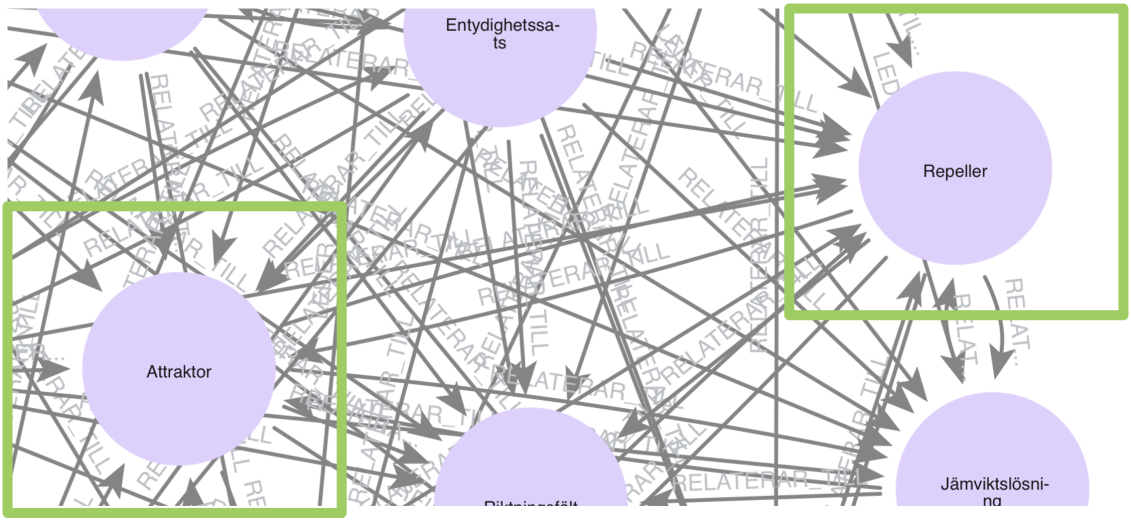
(d) Ytterligare isolerade öar

Figur 9: Intressanta områden i en graf genererad med stepwise prompting

Figur 9 visar ett antal intressanta områden från den graf som genererats med stepwise prompting. I systempromptet ges instruktionerna att dubbelrelationer inte ska skapas, vilket framgår från figur 9a att modellen haft svårt att följa. Figurerna 9b - 9d visar att modellen genererat grafen med ett flertal isolerade öar. En initial bedömning baserat på sunt förnuft kan resonera sig fram till att två av de isolerade öarna som visas i figur 9b, den som berör växande funktioner och den som berör avtagande funktioner, bör vara sammankopplade då de beskriver 'två sidor av samma mynt'. Det går även att resonera sig fram till att begreppen 'polynomdivision' borde ha en relation till begreppet 'polynom' - något som framgår inte finns i figur 9d, trots att de till och med tillhör samma isolerade ö.



Figur 10: Urklipp av en genererad graf från en kurs i transformmetoder



Figur 11: Urklipp av en genererad graf från en kurs i transformmetoder, med orden 'repeller' och 'attraktor' markerade

Figur 10 och 11 visar urklipp från en annan graf genererad från en kurs i transformmetoder

som exemplifierar ytterligare två förekommande fenomen i graferna som genererades med stepwise prompting. I figur 10 är det tydligt att språkmodellen inte har följt instruktionerna gällande antalet relationer som får skapas, där ett tydligt krav var att maximalt en relation får skapas mellan varje begreppspar. Antalet relationer mellan begreppsparen överstiger detta med råge. Figur 11 visar ett exempel på där 'nonsens'-ord inkluderats som begrepp. I detta fall handlar det om 'repeller' och 'attraktor', som efter närmare granskning kunde spåras till verben 'repellera' och 'attrahera' från teorikapitlen. Notera att graferna som illustreras ovan är *exempel* som illustrerar vanligt förekommande företeelser som även uppstod i andra grafer.

4.2 Prompt chaining

Nedan redovisas de prompter som använts i de olika stegen i prompt chaining-flödet, och sedan de kunskapsgrafer som genererats med metoden.

4.2.1 Promptgenerering

Promptet som användes i den AI-agent som extraherade nyckelord ses nedan i exempel 4.

```
# Prompt: Extrahera matematiska nyckelbegrepp

Du ska extrahera nyckelbegrepp från matematiska universitetstexter.
Följ dessa instruktioner exakt:

## Basregler
1. Extrahera specifika matematiska begrepp som är viktiga för ämnet
2. Inkludera begrepp från rubriker
3. Sortera begrepp i den ordning de dyker upp i texten
4. Presentera resultatet som en enkel txt-lista, ett begrepp per rad
5. **Inkludera rätt nivå av begrepp**: Extremt grundläggande begrepp som "addition"
inkluderas INTE, men universitetsrelevanta begrepp som "derivata", "integral" eller
"funktion"
SKA inkluderas om de är centrala i texten och NÄMNS EXPLICIT.
6. Du FÅR INTE extrahera begrepp som INTE finns i texten.

## KRITISKA REGLER - MÅSTE FÖLJAS
1. **INGA DUBLETTER TILLÅTS**: Samma begrepp får endast visas EN gång oavsett
grammatisk form
   - Singular/plural (t.ex. "koordinat"/"koordinater") = SAMMA BEGREPP
   - Bestämd/obestämd form (t.ex. "derivata"/"derivatan") = SAMMA BEGREPP
   - Verb/substantiv (t.ex. "projicera"/"projektion") = SAMMA BEGREPP
   - Olika prepositioner (t.ex. "projektion på"/"projektion av") = SAMMA BEGREPP

2. **Dela upp sammansatta begrepp**:
"Gränsvärden och kontinuitet" → "Gränsvärde", "Kontinuitet"

3. **Bevara sammanhängande begrepp**: "Derivatans definition" är ETT begrepp

4. **Om input inte ges i textformat/markdown-format, utan exempelvis är en länk som
leder till
en bild som du inte kan extrahera text från, ska du inte hitta på något utan isåfall
ange att
'Inputen gavs inte i ett textformat jag inte kunde tyda, inga nyckelbegrepp har
därför extraherats'.

5. Extrahera INTE begrepp som INTE FINNS MED I TEXTEN!!!!

## Process
1. Identifiera alla potentiella begrepp
2. Eliminera alla grammatiska varianter och behåll endast EN variant av varje begrepp
3. Dubbelkolla särskilt för singular/plural-dubletter och verb/substantiv-dubletter

## Exempel
Text: "Projektion på vektor. Projektionen av en vektor. Man kan projicera en vektor..."

FELAKTIGT resultat:
Projektion på vektor
Projektion av en vektor
Projicera en vektor

KORREKT resultat (välj EN variant):
Projektion på vektor

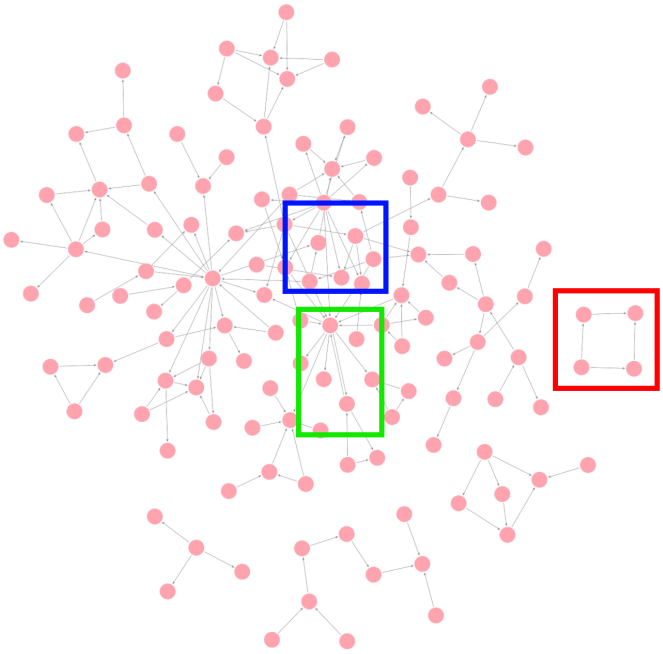
## SLUTKONTROLL
Gå igenom din lista en sista gång för att verifiera att inga
grammatiska varianter återstår.
```

Exempel 4: Systempromptet som användes för att extrahera matematiska nyckelbegrepp

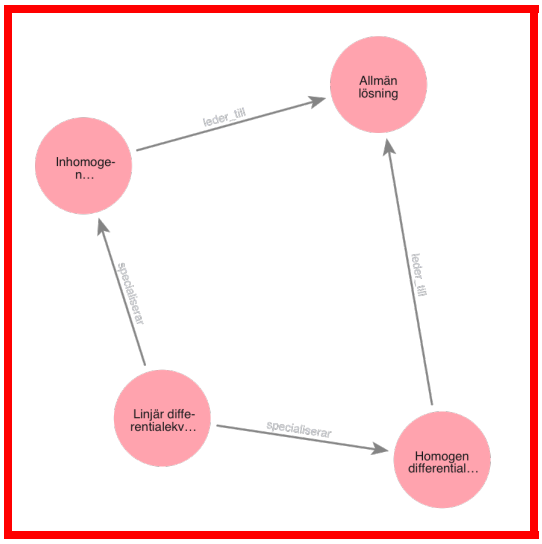
För att se de övriga individuella prompts till de olika agenterna, se bilaga D

4.2.2 Kunskapsgrafer genererade med prompt chaining

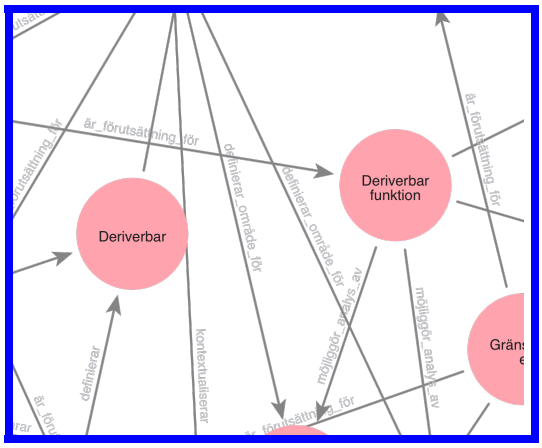
Ett exempel på en kunskapsgraf som genererades på en hel kurs visas i figur 12, där en kurs i envariabelanalys användes. De färgade rutorna förklaras närmare i figur 13a - 13c.



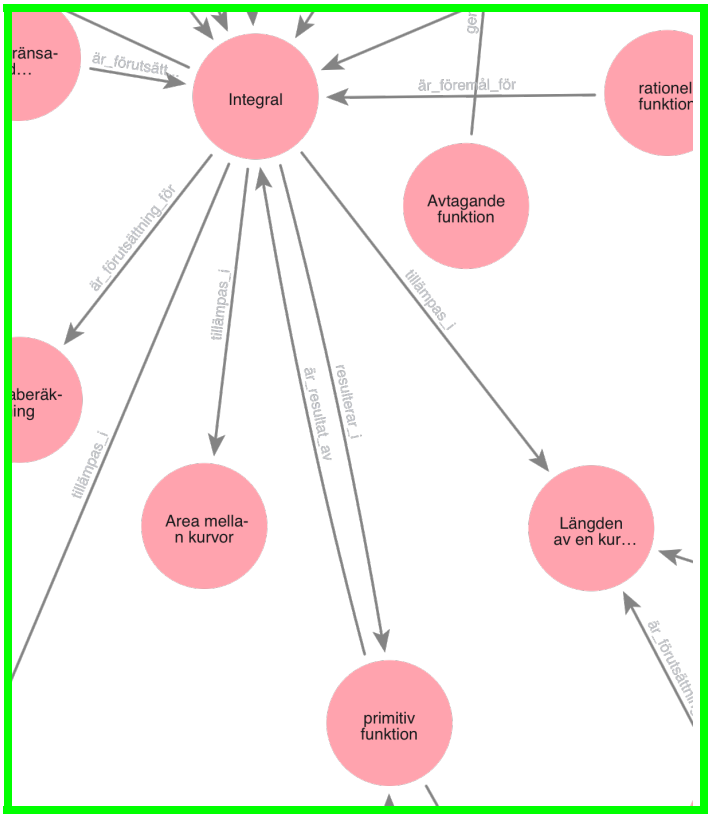
Figur 12: Genererad graf för kursen Envariabelanalys med prompt chaining



(a) Isolerad ö med begrepp kopplade till differentialekvationer



(b) Två separata noder med begreppen ’deriverbar’ och ’deriverbar funktion’



(c) Dubbelrelation mellan integral och primitiv funktion

Figur 13: Intressanta områden i en graf genererad med prompt chaining

Som tidigare sett förekom problem med isolerade öar redan i de grafer som genererats med stepwise prompting. Denna företeelse går även att observera i de prompt chaining-genererade graferna. I figur 13a går det att se en isolerad ö med fyra begreppsnoder in-

om området differentialekvationer. Endast en grundläggande förståelse för envariabelanalys krävs för att veta att differentialekvationer har en direkt stark koppling till derivata. Att dessa begreppsnoder inte har någon koppling i grafen är ett tecken på att modellen säkerligen har missat att identifiera flera relationer som varit högst rimliga att identifiera. Figur 13b visar på hur två separata begreppsnoder extraherats - en som heter 'deriverbar' och en som heter 'deriverbar funktion'. Den mest troliga förklaringen till detta är att detta begrepp förekommer i olika kapitel i olika sammanhang - och att det i integrationssteget inte har identifierats att dessa har med varandra att göra. Man hade kunnat argumentera för att de beskriver samma sak, och då borde språkmodellen i integrationssteget ha slagit ihop dessa begrepp. Man kan även argumentera för att de bör särskiljas som olika begrepp, men då bör det åtminstone finnas en relation mellan dessa begrepp, vilket modellen inte har skapat. I figur 13c går det att se att modellen, likt i fallet med stepwise prompting, fortfarande skapar dubbelriktade relationer trots instruktioner i promptet att inte göra detta.

4.3 Tester av nyckelordsextrahering

Nedan följer resultaten från testerna av nyckelordsextraheringen. Först redovisas de kvantitativa resultaten, där fokus ligger på att identifiera skillnader i stabilitet för de olika inställningarna. Sedan redovisas den kvalitativa delen, där en stickprovsanalys av utvalda nyckelord sker.

4.3.1 Kvantitativ analys

I figurerna nedan visas resultaten från analyser av nyckelordsextrahering från följande kapitel om derivatans definition från en kurs i envariabelanalys, som kan ses nedan i exempel 5.

Definition: Derivatan av en funktion f i punkten $x = a$ definieras som

$$f'(a) = \lim_{h \to 0} \frac{f(a+h) - f(a)}{h}$$

om detta gränsvärde existerar.
Funktionen f sägs då vara *deriverbar* i $x = a$. Annars sägs det att f är oderiverbar i $x = a$ eller att derivatan av f inte existerar i $x = a$.

Sats: Om f är deriverbar i $x = a$, är f *också* kontinuerlig i $x = a$.

Att komma ihåg:

1. Gränsvärdet $\frac{f(a+h) - f(a)}{h}$ då $h \to 0$ existerar om och endast om

   $$\lim_{h \to 0^+} \frac{f(a+h) - f(a)}{h} = \lim_{h \to 0^-} \frac{f(a+h) - f(a)}{h} = C$$

   där C är ett tal.

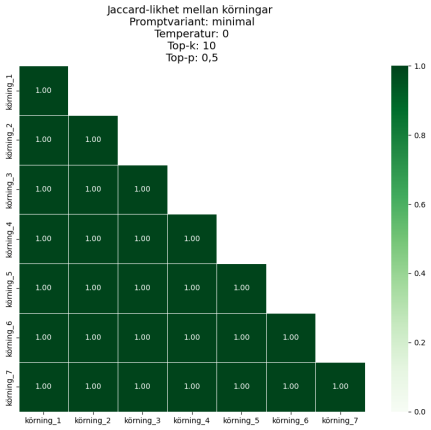
2. Om f är definierad endast på ett slutet intervall $[a, b]$, existerar inte derivatan av f i ändpunkterna $x = a$ och $x = b$.

3. Om f inte är definierad eller kontinuerlig i $x = a$, kan f omöjligen ha derivatan i $x = a$.

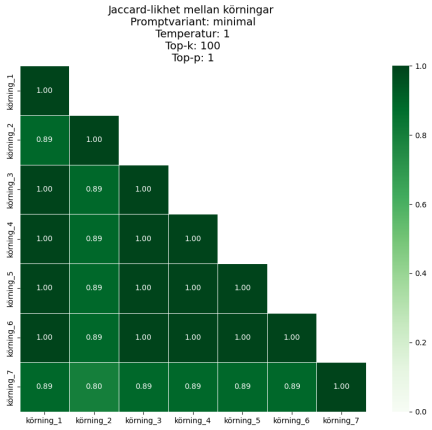
4. Ett tentamenspopulärt tecken för oderiverbarhet av en funktion i $x = a$ är att funktionens graf inte är glatt i $x = a$. Tänk på grafen för kändisen $f(x) = |x|$ som inte är deriverbar i $x = 0$.

Exempel 5: Teorikapitel från Kollins databas om derivatans definition, hämtat från en kurs i envariabelanalys

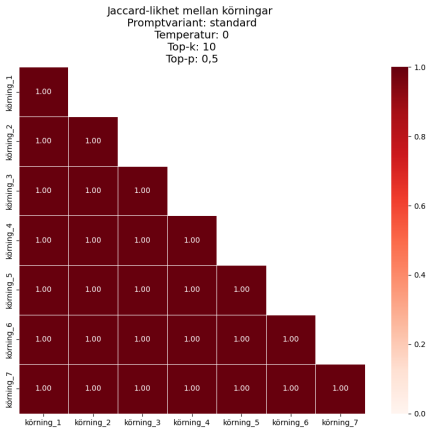
Figur 14 visar hur lika de 7 körningarna för varje hyperparameterkombination är med avseende på Jaccardlikhet. Figur 14a visar likhetsindex för körningar med temperatur = 0, top-k = 10, top-p = 0,5, alltså inställningar som förväntas ge små eller inga variationer i resultat. Figur 14b visar likhetsindex för körningar med temperatur = 1, top-k = 100, top-p = 1, alltså inställningar som förväntas ge högre variation. Båda dessa figurer visar körningar för den minimala varianten av promptet. Figur 14c och 14d visar resultaten från samma hyperparameterinställningar med standardvarianten av promptet, och figur 14e och 14f för det detaljerade promptet. I varje figur förekommer körningarna både på x-axeln och y-axeln, och i deras skärningspunkt anges värdet för deras likhet. Ett likhetsindex på 1 innebär att körningarna gav identisk output, medan ett likhetsindex på 0 innebär att inga nyckelord i de två givna körningarna var samma. Naturligt så har körningarna alltid ett likhetsindex på 1 där de 'skär sig själva'. Färgkodningen i figur 14 fyller endast funktionen att visuellt kunna särskilja data från de olika promptvarianterna.



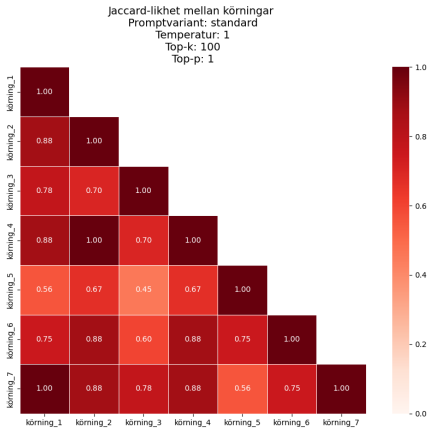
(a) Minimalt prompt, temperatur = 0, top-k = 10, top-p = 0,5



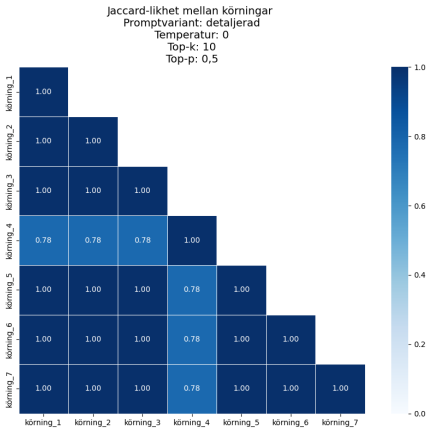
(b) Minimalt prompt, temperatur = 1, top-k = 100, top-p = 1



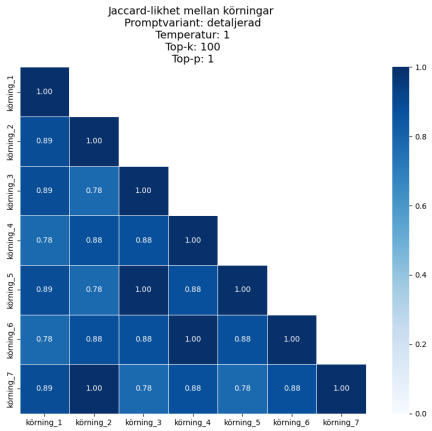
(c) Standardprompt, temperatur = 0, top-k = 10, top-p = 0,5



(d) Standardprompt, temperatur = 1, top-k = 100, top-p = 1

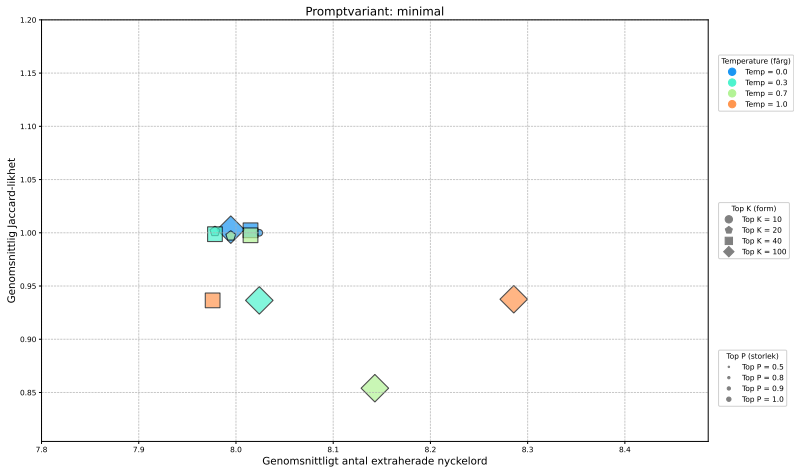


(e) Detaljerat prompt, temperatur = 0, top-k = 10, top-p = 0,5

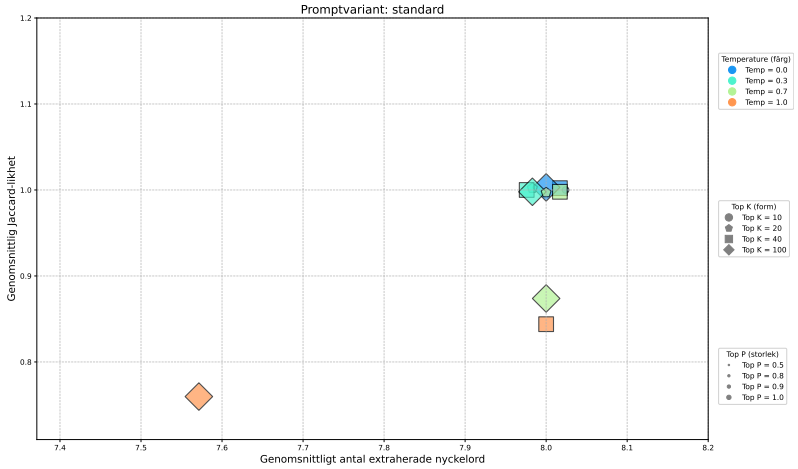


(f) Detaljerat prompt, temperatur = 1, top-k = 100, top-p = 1

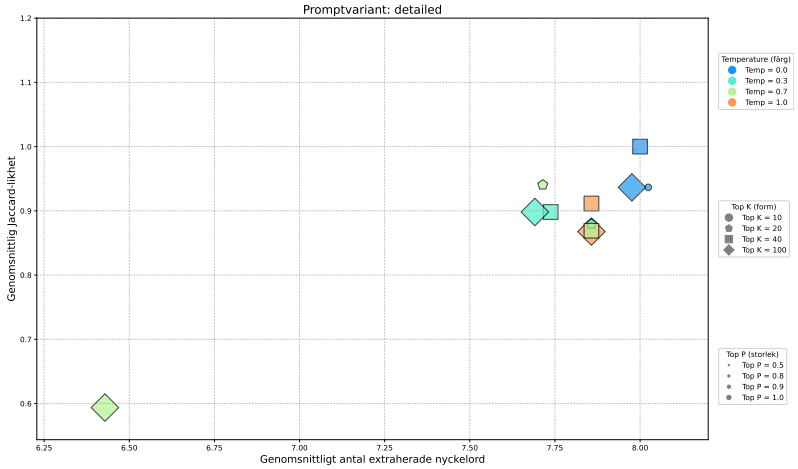
Figur 14: Likhetsindex för sex olika kombinationer av prompttyper och hyperparameterinställningar.



(a) Genomsnittlig Jaccard-likhet och extraherade nyckelord för den minimala promptvarianten



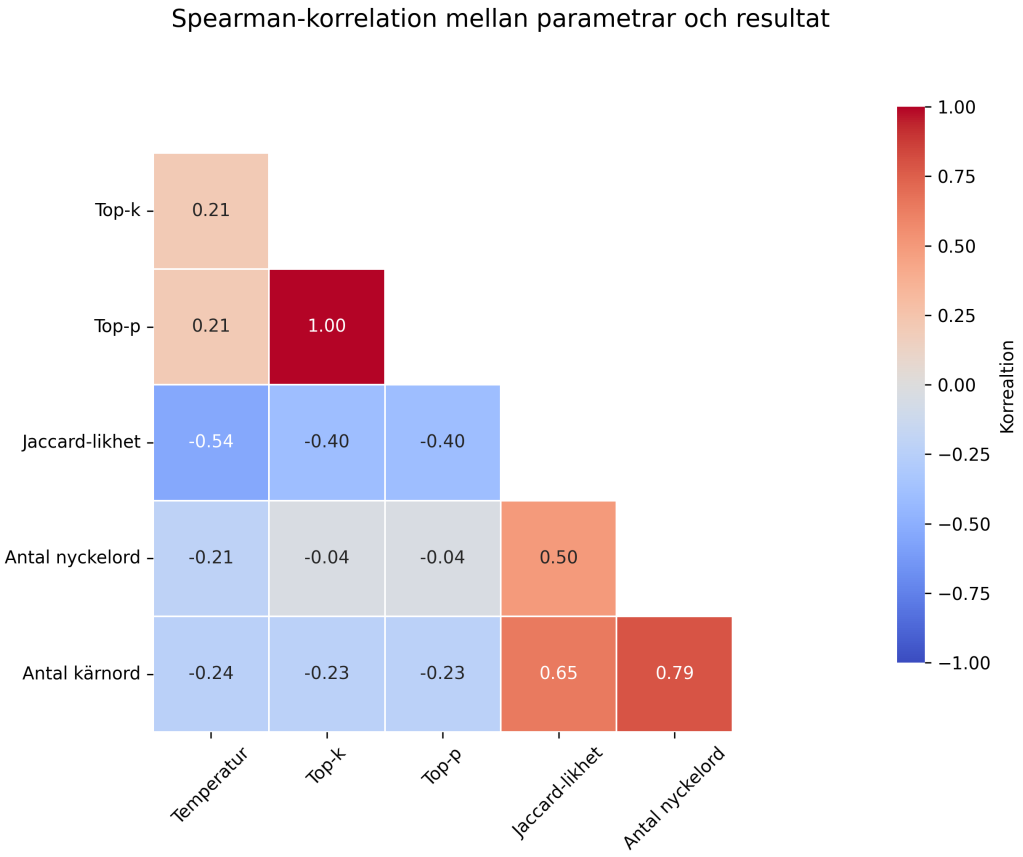
(b) Genomsnittlig Jaccard-likhet och extraherade nyckelord för standard-promptvarianten



(c) Genomsnittlig Jaccard-likhet och extraherade nyckelord för den detaljerade promptvarianten

Figur 15: Jämförelse av stabilitet för olika promptvarianter

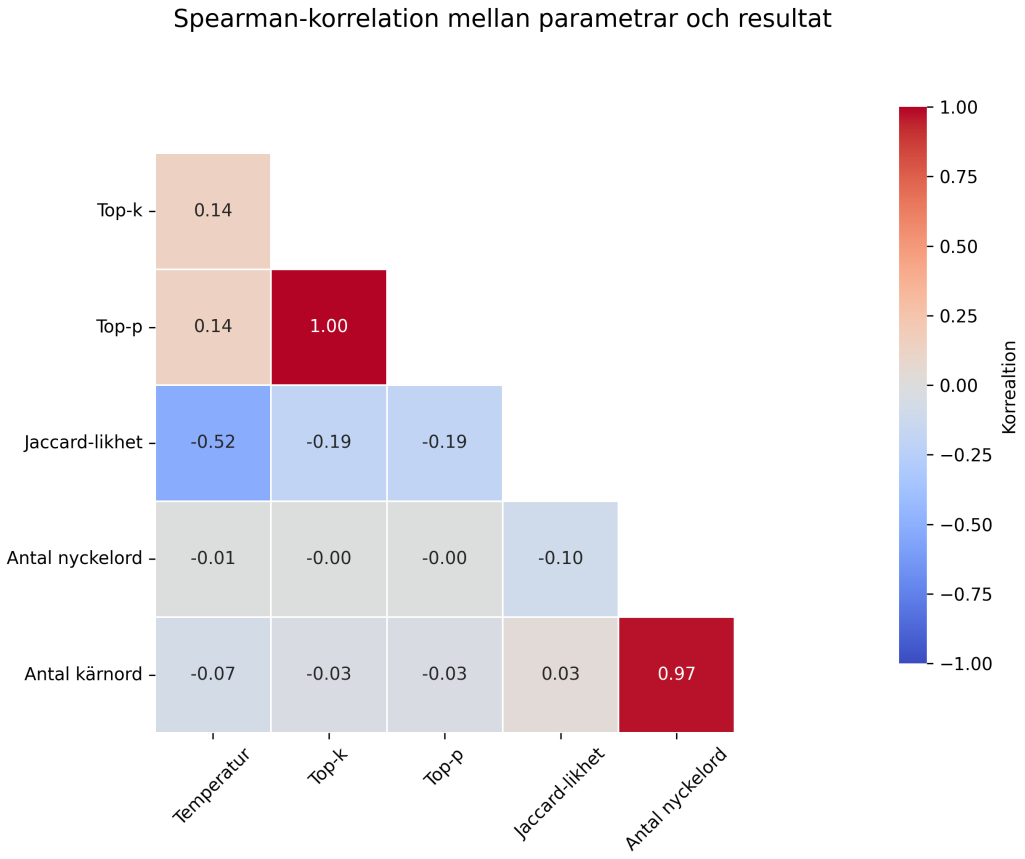
I figur 15a-15c visas den genomsnittliga jaccardlikheten för varje kombination av hyperparametrar. Varje delfigur visar resultatet från en promptvariant, där figur 15a gäller för den minimala promptvarianten, 15b gäller för standardvarianten, och figur 15c för den detaljerade varianten. Figurens x-axel visar hur många nyckelord i snitt som extraherats per hyperparameterkombination, och y-axeln visar den genomsnittliga Jaccardlikheten för kombinationen i fråga. Datapunkterna har olika färg, form och storlek, där färgen representerar vilken temperatur som använts, formen representerar värdet på top-k, och storleken representerar värdet på top-p. En datapunkt som är mörkblå, har formen romb och den största storleken, representerar alltså en kombination med t=1, top-k=100, top-p=1. Figurerna visar data från kapitlet om derivatans definition som redovisats tidigare. För alla dessa tre promptvarianter verkar det gemensamma vara att körningar med höga temperaturer (de med mörkblå färg) klustrar sig kring högt likhetsindex.



Figur 16: Korrelation mellan parametrar och resultat för kapitlet om derivatans definition

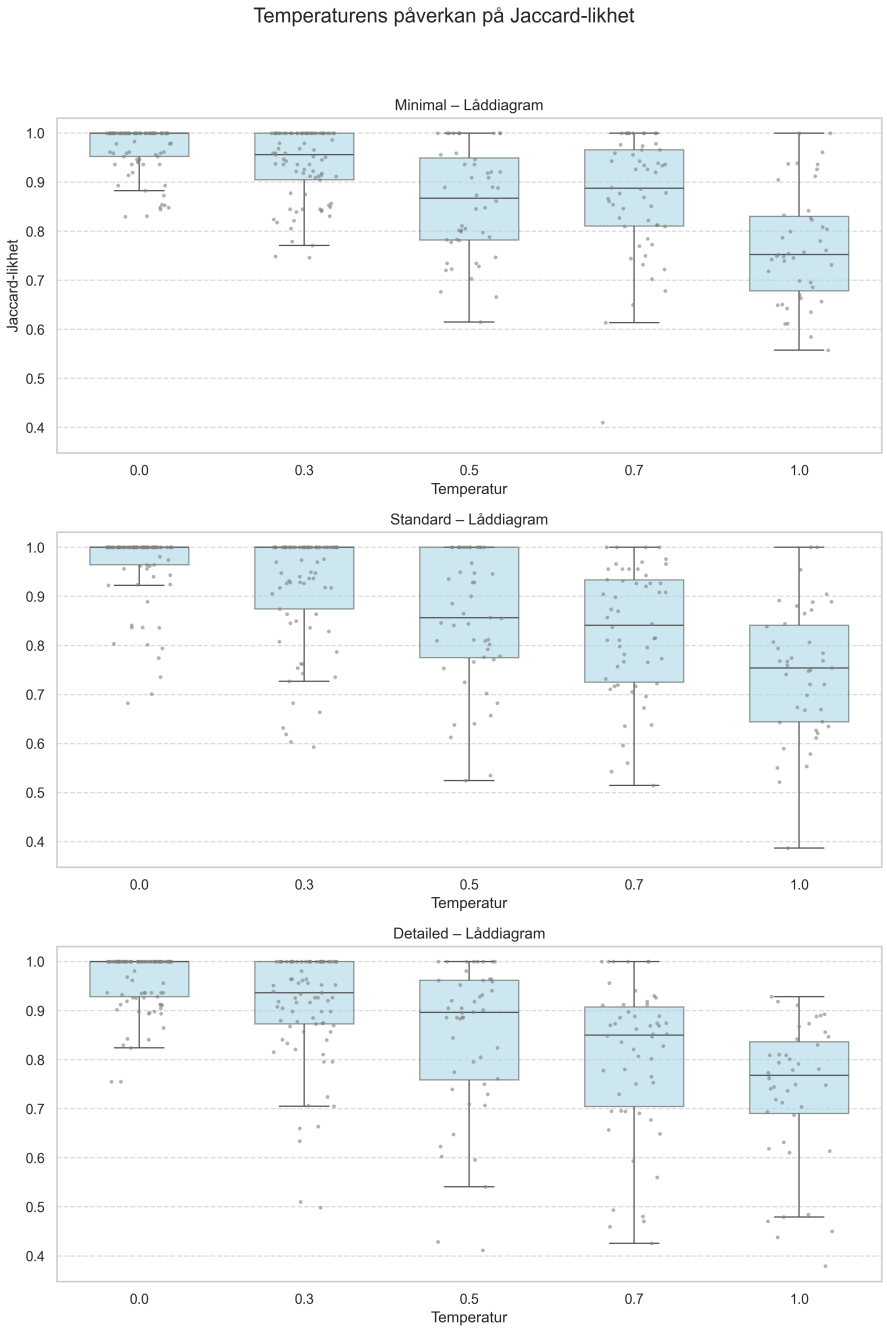
För att undersöka huruvida de olika hyperparametrarna hade någon faktiskt signifikant påverkan på hur stabil utdata språkmodellerna gav i nyckelordsextraheringen gjordes en korrelationsanalys. Med stabil i denna bemärkelse menas likhet mellan körningar med samma inställningar. Figur 16 visar de olika parametrarnas enskilda påverkan på olika dimensioner av resultatet. Korrelationerna i figur 16 är baserade på de extraherade nyckelorden från kapitlet om derivatans definition, och det går att utläsa att den parameter som har störst korrelation med Jaccardlikheten är temperaturen, följt av top-k och sedan

top-p. För att avgöra om de siffrorna är representativa gjordes samma analys på hela det undersökta materialet, som kan ses i figur 17.



Figur 17: Korrelation mellan parametrar och resultat för alla 21 analyserade kapitel

Efter att korrelationsanalys har gjorts på nyckelordsextrahering från alla 21 undersökta teorikapitel, är det tydligt att den parameter som har störst korrelation med Jaccardlikheten är temperaturen, medan top-k och top-p verkar ha mindre betydelse. Detta kan ses i figur 17. För att få en tydligare bild av hur Jaccardlikheten varierar inom de olika temperaturinställningarna visualiseras detta i figur 18. Där representeras varje ’körningsgrupp’ av en datapunkt, där temperaturen plottas ut på x-axeln och den genomsnittliga Jaccardlikheten plottas ut på y-axeln, uppdelat för varje promptvariant. Tabell 3 visar den första temperatur där signifikant skillnad i Jaccard-likhet observeras jämfört med t = 0, för varje promptvariant. För den minimala och detaljerade promptvarianten ses en signifikant skillnad redan vid temperaturen 0,3, medan standardvarianten inte visar någon signifikant skillnad mellan t = 0 och t = 0,3, men däremot mellan t = 0 och t = 0,5.

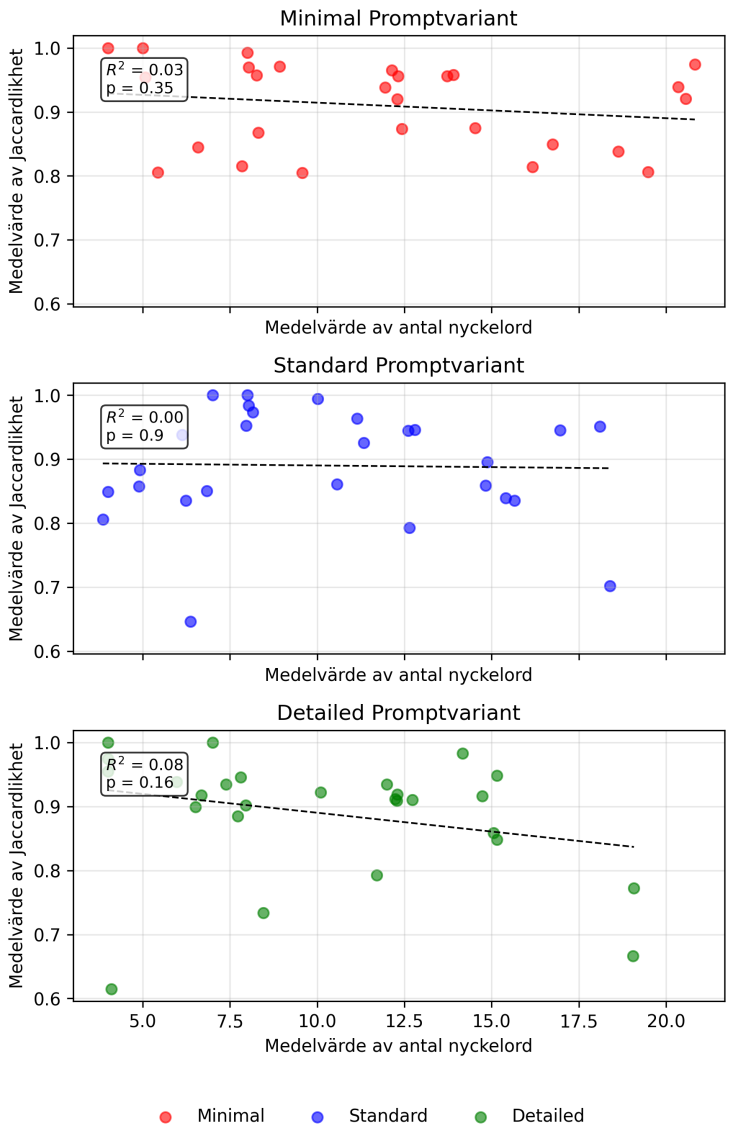


Figur 18: Temperaturens påverkan på Jaccardlikheten

Tabell 3: P-värden från Dunn's test för jämförelse med t = 0.0 per promptvariant.

Promptvariant	**Temperatur (t)**	**p-värde**	**Signifikant**
Minimal	0.3	0.0471	Ja
Standard	0.3	0.1123	Nej
Standard	0.5	0.0000	Ja
Detaljerad	0.3	0.0181	Ja

En annan dimension som undersöktes var huruvida likheten mellan körningarna påverkades av hur många nyckelord som extraherades. Utgångsantagandet var att det fanns en möjlighet att större variation uppkommer när det finns fler nyckelord inom varje kapitel. Detta analyseras med regressionsanalys i figur 19.



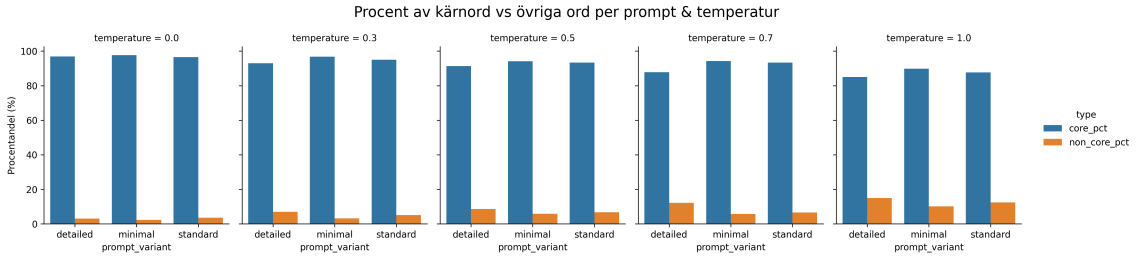
Figur 19: Regressionsanalys av antal nyckelord och Jaccardlikhet

Som kan ses i figur 19 verkar det inte finnas något tydligt samband mellan antal extraherade nyckelord och likhet mellan körningarna för någon av promptvarianterna. Ett möjligt samband kan urskiljas för den detaljerade promptvarianten, men R^2-värdet på 0,08 i kombination med p = 0,16, som indikerar att sambandet inte är statistiskt signifikant, motsäger detta. Att det inte finns någon tydlig korrelation skulle kunna tyda på att språkmodellen klarar av att extrahera nyckelord från längre teorikapitel (då flera nyc-

kelord har en implicit koppling till större källmaterial) med samma stabilitet som från kortare kapitel. Språkmodellen verkar vara tillräckligt robust för att hantera den längd på textkapitel som testmaterialet utgjorde. Hur bra den hade presterat, med avseende på likhet mellan körningar, för betydligt längre textkapitel har inte inom ramen för detta arbete varit relevant att undersöka.

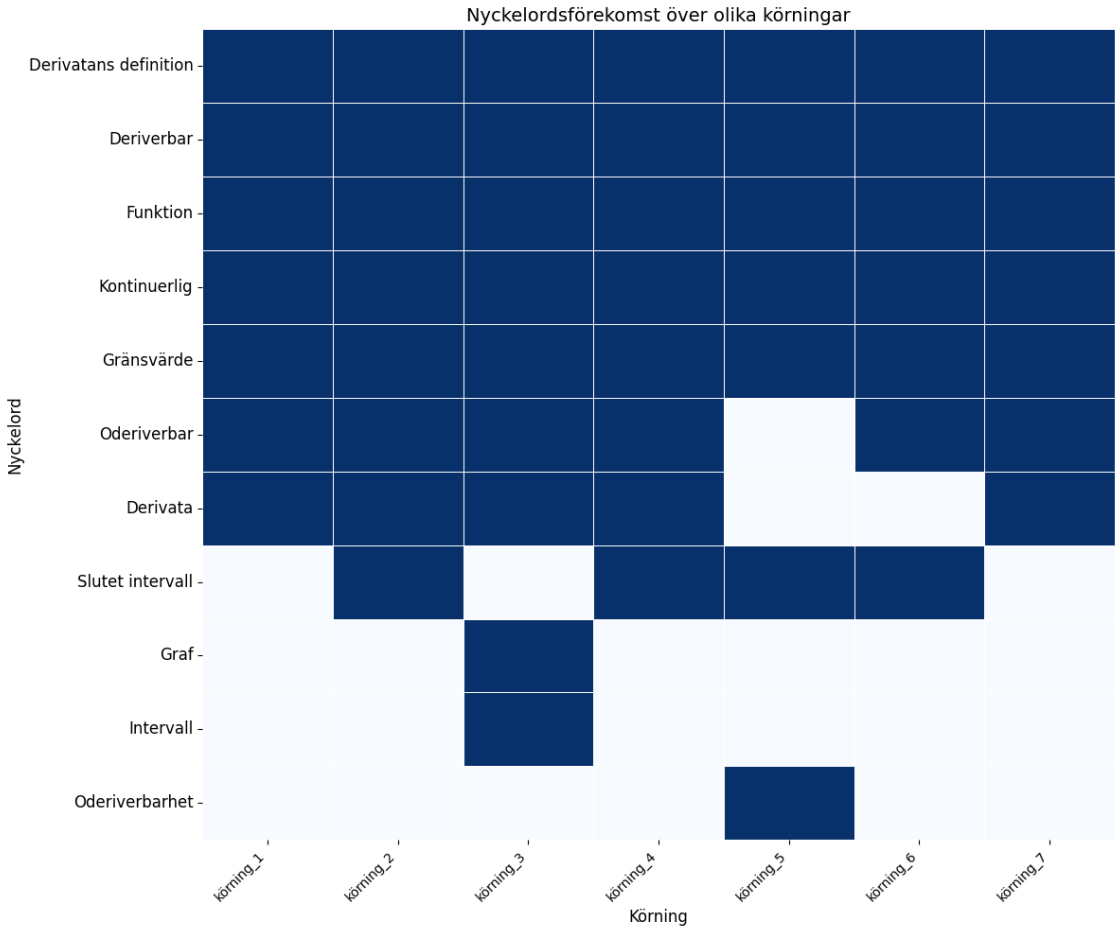
4.3.2 Kvalitativ analys

I den kvantitativa analysen kunde information fås om hur många nyckelord som extraherades per körning, hur lika körningarna var varandra, samt vilka hyperparametrar som verkade ha störst inverkan på stabiliteten i outputen. Den datan säger dock ytterst lite om de faktiska nyckelorden som extraherats. Ett mått som användes i den kvantitativa analysen var andelen kärnord i förhållande till andelen övriga ord. Ett ord definierades som ett kärnord om det förekom i över 70% av körningarna för en viss hyperparameterkombination. Eftersom att varje kombination testades för 7 körningar klassades alltså ett ord som ett kärnord om det förekom i 5 eller fler av dessa. I figur 20 går det att se att andelen nyckelkärnord minskar när temperaturen ökar, något som även kan härledas av de figurer som tidigare presenterat hur väl temperaturen har korrelerat med Jaccardlikheten. Att Jaccardlikheten är lägre innebär att fler unika ord förekommer som inte är en del av alla körningar. Viktigt att poängtera är dock att andelen icke-kärnord vid högre temperaturer var under 20% för alla promptvarianter, vilket innebär att det även för de inställningar som har gett högst variation, fanns en solid uppsättning kärnord.



Figur 20: Kärnord jämfört med övriga ord

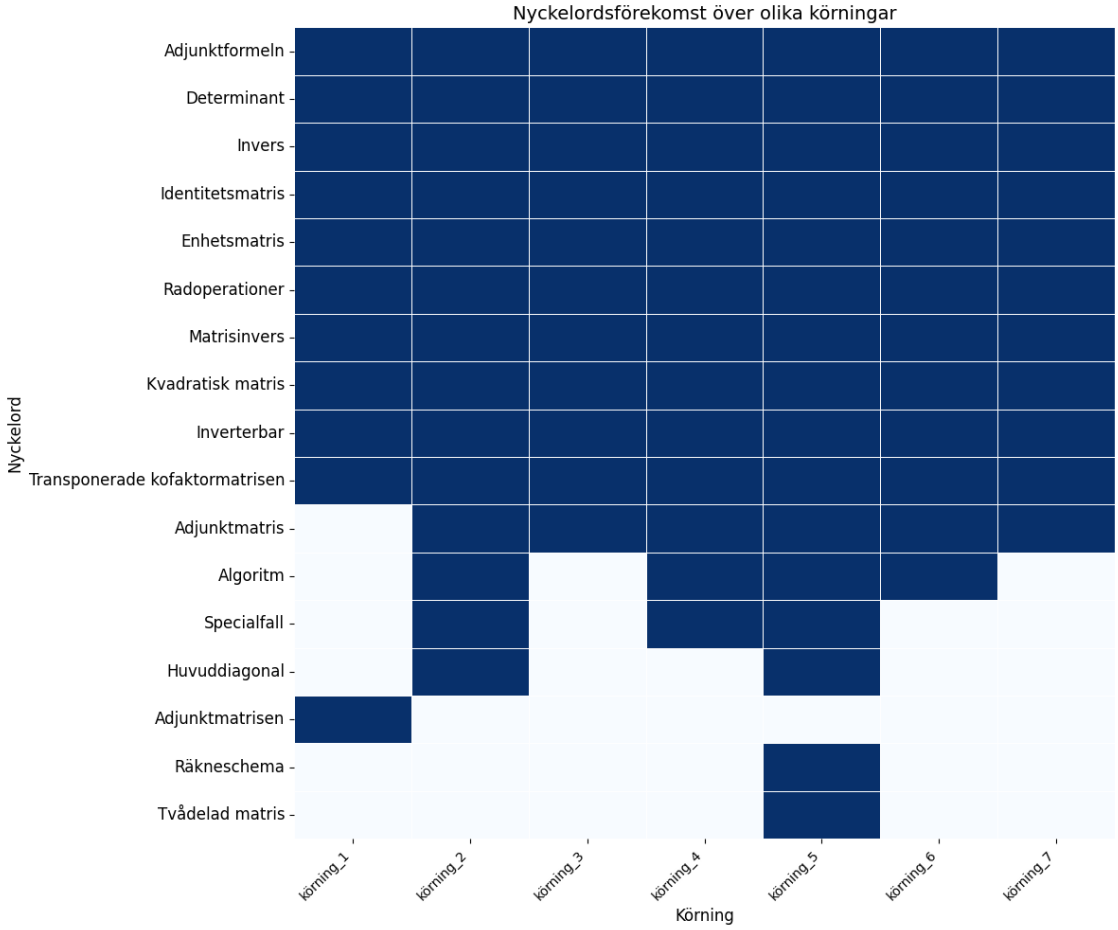
Nedan redovisas ett par exempel från körningar med de inställningar som genererat störst variation av nyckelord. I figurerna går det att se vilka nyckelord som extraherats från respektive körning. Orden i fråga är utskrivna i vänsterspalten och om ordet förekommer i den körning som anges på figurens x-axel markeras detta med en ifylld blå ruta. Orden från figur 21 visar de ord som extraherats från kapitlet om derivatans definition (se 4.3.1).



Figur 21: Extraherade nyckelord från kapitel om derivatans definition, temperatur = 1

Att endast säga att körningarna inom en viss hyperparameterkombination är lika alternativt olika varandra säger bara något begränsat om vad den faktiska outputen består av. I figur 21 går det att se att en viss del av variationen kan förklaras av grammatik. Exempelvis går det att se att körning 5 har extraherat ordet ’oderiverbarhet’ medan de andra körningarna har extraherat ’oderiverbar’. Båda ord refererar till samma matematiska koncept, och ingen av körningarna har extraherat båda de grammatiska varianterna. 5 av 7 körningar extraherar termen ’derivata’ från kapitlet, medan övriga två körningar inte gör det. I ett kapitel som handlar om derivatans definition hade det gått att argumentera för att ordet ’derivata’ är ett nyckelbegrepp, även om ’derivatans definition’ som sammansatt begrepp också har extraherats. Däremot är ’funktion’ ett ord som har extraherats från alla körningar, som utan sitt sammanhang skulle kunna anses vara något intetsägande. I kontexten av det aktuella kapitlet så kommer ordet ’funktion’ från sammanhanget ’deriverbar funktion’, varpå det kan ifrågasättas att ’funktion’ klassificeras som ett ensamt nyckelbegrepp. Körning 3 extraherar även orden ’graf’ och ’intervall’. Dessa ord är visserligen matematiska men det kan ifrågasättas hur relevanta de är som fristående nyckelord inom ramen för matematiska kurser på universitetsnivå. Begreppet ’slutet intervall’, som extraherats i 4 av 7 körningar, är mer talande för den aktuella kontexten. Ett slutet intervall är visserligen ett typ av intervall, men i till exempel körning 3 där ’intervall’ extraheras men

inte ’slutet intervall’, går det med viss säkerhet att säga att viktig information går förlorad. Om båda begreppen extraheras hade detta kunnat förklaras med en relation som skapas i de senare stegen som förklarar att ’slutet intervall’ är en typ av ’intervall’. Gällande ordet ’graf’ som identifierats som ett nyckelord i en av körningarna kan det härledas till formuleringen i teorikapitlet som lyder: ’Ett tentamenspopulärt tecken för oderiverbarhet av en funktion i x=a är att funktionens graf inte är glatt i x=a.’ Att språkmodellen här väljer att klassificera ’graf’ som ett nyckelord är anmärkningsvärt då det framgår av kontexten att graf som självständig term inte förklarar något av relevans gällande just detta område. Hade ’graf’ som koncept beskrivits i kapitlet med förklarande termer hade det varit mer motiverat att inkludera ordet i listan över extraherade begrepp.



Figur 22: Extraherade nyckelord från kapitel om matrisinvers, temperatur = 1

Fler exempel på de typer av begrepp som extraheras i körningar med hög temperatur kan ses i figur 22, som visar ord extraherade från ett kapitel om matrisinverser från en kurs i Linjär algebra. Kapitlet kan ses i sin helhet i bilaga F. Här förekommer också variation med avseende på grammatiska varianter såsom ’Adjunktmatrisen’ istället för ’Adjunktmatris’ i körning 1. På samma sätt som den matematiska relevansen av begreppen ’graf’ och ’intervall’ på den aktuella nivån kan ifrågasättas, kan i detta fall orden ’algoritm’ och ’specialfall’ sättas i samma kategori. Utifrån kapitlet går det att utläsa att ’algoritm’ är en underrubrik som beskriver en algoritm för att bestämma matrisinversen, likaså är ’special-

fall' en rubrik som beskriver ett specialfall av adjunktformeln. Dessa ord har alltså ingen matematisk relevans i sin egenhet, utan måste ses i ett större sammanhang som inte fångas av nyckelordsextraheringen när den får så mycket 'frihet' som i temperatur = 1. Intressant är även begreppet 'räkneschema' i körning 5 som i kapitlet antas vara ett etablerat begrepp där ett specifikt räkneschema beskrivs. Det kan då ifrågasättas att endast 'räkneschema' extraheras utan någon närmare specifikation av vilket räkneschema som åsyftas.

5 Diskussion

I detta kapitel diskuteras insikter från resultaten, både från stepwise prompting och prompt chaining som tillvägagångssätt, samt de kvantitativa och kvalitativa resultaten från nyckelordsanalysen. Även möjliga alternativa tillvägagångssätt som hade kunnat användas under processens gång diskuteras.

5.1 Stepwise prompting & prompt chaining

Det svåraste med att dra några betydelsefulla slutsatser från hur väl de två promptingteknikerna presterar när det gäller att skapa meningsfulla grafer är att det inte finns något tydligt facit att jämföra resultaten mot. Det finns ingen referensgraf för exempelvis kursen Envariabelanalys som har kunnat fungera som riktmärke. Då det inte har kunnat kvantifieras hur bra dessa tekniker har utfört sin uppgift, har istället ett mer intuitionsbaserat tillvägagångssätt tagits till resultattolkningen.

För det första är det värt att notera att de figurer som presenteras i 4.1.2 och 4.2.2 är *exempel* på de grafer som har genererats med de givna metoderna. Under arbetets gång har flertalet grafer skapats, och dessa har alla skiljt sig åt mellan varandra, trots att de har gjorts för samma kursmaterial och med samma inställningar. Detta är givet då total determinism inte på något sätt är en förväntad egenskap för stora språkmodeller. Dessutom är det svårt att 'klumpa samman' dessa grafer till något betydelsefullt. Visst hade man kunnat beräkna ett medelvärde av antalet noder och antalet relationer - men frågan är vad de resultaten hade berättat utan att komplettera detta med en gedigen analys av noderna och relationernas innehåll.

Istället, som exemplifieras av figurerna i 4.1.2 och 4.2.2 går det att peka på några intressanta vanligt förekommande fenomen i graferna och teoretisera om deras orsaker. Något som identifierades som en potentiell risk med prompt chaining var att det skulle skapas isolerade öar i grafen då kapitlen behandlades ett åt gången istället för att matas in allihopa samtidigt, och därmed missa beroenden mellan kapitlen. Isolerade öar förekom visserligen i resultaten från prompt chaining, men det gjorde det även med stepwise prompting. Det går även att i resultaten för prompt chaining se att liknande begrepp extraheras från olika kapitel utan att identifieras som dubbletter eller ens ha en relation till varandra (se figur 13b). Trots att det finns felkällor som liknar varandra både i resultaten från stepwise prompting och prompt chaining, finns det fördelar med att använda prompt chaining. Genom att dela upp processen i en modulär struktur istället för att försöka producera en hel graf i ett massivt prompt, möjliggör detta att kunna felsöka individuella steg. Denna möjlighet bör i ett realistiskt scenario väga tyngre än det faktum att prompt chaining kan leda till högre kostnader på grund av flera API-anrop till språkmodellen.

Ett ytterligare problem, som en potentiell lösning till diskuteras i 5.3, vilket kan härledas till båda metoderna, är att mängden input tokens har en betydelse för hur väl modellen klarar av uppgiften. I prompt chaining-modellen blir storleken på input större och större

för varje iteration av kapitelintegration, i och med att det nya kapitlet samt hela den befintliga grafen ges som input till språkmodellen. Att detta skulle ha en eventuell negativ effekt på utdatans kvalitet går hand i hand med de resultat som presenteras i studien *Same task, more tokens*, som tidigare redovisats i kapitel 2.1.3. Att detta har relevans kan även motiveras av att figur 7, som visar grafen genererad på endast ett kapitel, inte innehåller några isolerade öar eller dubbelriktade relationer, som förekom då flera kapitel användes.

5.2 Insikter från testerna av nyckelordsextrahering

Delen av arbetet som behandlar nyckelordsextraheringen gav upphov till en rad diskussionspunkter, som behandlas nedan. Dessa är uppdelade utefter insikter från den kvantitativa analysen respektive den kvalitativa analysen.

5.2.1 Kvantitativ analys

Utifrån de tester som gjordes där de olika hyperparameterkombinationerna använts för att undersöka vilka inställningar som kan generera en stabil output finns det en huvudsaklig lärdom vilket är att temperaturen är den faktor som haft störst påverkan på hur lik utdatan blivit då man anropat språkmodellen med samma instruktioner i flera körningar. Att temperaturen hade denna påverkan var ett väntat resultat, då temperaturen just är det som styr hur stor sannolikhet det är att ett visst ord blir valt som nästa token - låg temperatur ger en 'snål' sannolikhetsfördelning medan en hög ger det motsatta (se 2.1.4). Tanken bakom att göra denna typ av tester var att undersöka huruvida en språkmodell i denna kontext kan anses vara pålitlig i den mån att den skapar grafer på ett konsekvent sätt oberoende av hur många gånger grafen skapas. Målet bör alltså vara att en graf som skapas med en automatiserad process med stora språkmodeller som huvudsakligt stöd måste kunna ha någon form av förutsägbarhet för att kunna säkerställa att den är relevant och håller god kvalitet. Hyperparametrarna top-k och top-p visade sig inte ha någon signifikant inverkan på likheten mellan körningarna. Då top-k och top-p egentligen varierar samma sak - antalet tokens som modellen får välja mellan som nästa i utdatan - hade det varit en god idé att testa att variera dessa parametrar separat och inte båda samtidigt, som gjordes här. Att båda varierades samtidigt kan motiveras med att det inte gick att hitta någon tidigare studie där dessa lagts som filter ovanpå varandra, samt att det skulle bli beräkningsmässigt tungt att variera dessa separat då det hade lett till många fler hyperparameterkombinationer. Med facit i hand borde dock dessa parametrar varierats separat då det i analysen var svårt att utläsa en påverkan från någon av dem - hade de separerats hade det möjligen gått att tydligare utläsa eventuella mönster i de enskilda parametrarnas betydelse. Val av promptvariant hade viss betydelse, då tröskelvärdet för när temperaturförändring orsakade en signifikant skillnad i likheten var 0,3 för den minimala och detaljerade varianten, och 0,5 för standardvarianten. Detta kan indikera att språkmodellen fick för mycket frihet med de minimala instruktionerna, och eventuellt blev förvirrad av de längre, mer detaljerade instruktionerna. I praktiken skulle ett sådant tröskelvärde kunna användas om det önskas hitta en balans mellan determinism och att fånga in så många nyckelord som möjligt, utan att påverka stabiliteten signifikant. Inte heller verkar det ha varit så att antalet nyckelord

som extraherats - vilket får antas ha en korrelation med teorikapitlets längd - har haft någon påverkan på hur lika körningarna har varit. Detta bekräftas av figur 19 där det går att se att det inte finns något tydligt samband mellan antalet extraherade nyckelord och likheten i kapitlets körningar.

5.2.2 Kvalitativ analys

Att endast se till de kvantitativa resultaten skulle vara något intetsägande och inte ge en god helhetsbild av hur väl detta tillvägagångssätt sätt för att skapa kunskapsgrafer fungerar och är hållbart. Stabilitet i output bör vara en viktig komponent, men att ett visst antal körningar är lika varandra säger ingenting om själva innehållet i språkmodellernas utdata. Potentiellt hade en värdelös språkmodell kunnat extrahera totalt irrelevanta ord från en text - men så länge orden hade varit identiska hade den i den kvantitativa analysen fått en 'hög poäng'. För att ge ett extremt exempel hade en 'dålig' språkmodell potentiellt kunnat missförstå instruktionerna totalt och istället för att agera som en matematisk expert som ska extrahera matematiska nyckelord, agerat som en språkexpert som ska extrahera prepositioner. Modellen hade då kanske med säkerhet extraherat orden [i, på, under] i varje körning. Även om varje körning skulle extrahera exakt dessa ord skulle utdatan vara totalt värdelös i och med att de saknar matematisk relevans. Just detta extrema utfall är nog inte särskilt troligt, men fungerar som ett exempel för att visa att det i denna analys har varit av hög relevans att inte stirra sig blind på likhetsindexet som ett alltför tillförlitligt mått på vilken kvalitet graferna skapas med.

För att med någorlunda säkerhet kunna bedöma kvaliteten av de extraherade nyckelorden, användes ett kapitel från en kurs i envariabelanalys och ett kapitel från en kurs i linjär algebra. Dessa valdes då de ansågs vara tillräckligt grundläggande för att möjliggöra en rimlig kvalitetsbedömning av nyckelorden. De ansågs också vara representativa för fenomen som förekommit i andra kapitel. Som redan analyserats i kapitel 4.3.2 förklaras mycket av variationen inom kapitlet av grammatiska variationer av samma begrepp. Detta torde inte vara ett stort problem - ett ytterligare steg som normaliserar de extraherade begreppen och omformulerar dem efter en önskad grammatisk form är inte en omöjlig teknisk implementering. Det ska även poängteras att det har varit mycket ovanligt i testerna att grammatiska variationer har förekommit inom samma körning, utan det har i de allra flesta fall varit så att exempelvis 'matrisinvers' har förekommit i körning X och 'matrisinversen' har förekommit i körning Y. Detta är också en anledning till att likhetsindexet utgör ett ofullständigt mått i sig självt, och att den kvantitativa analysen har sina brister.

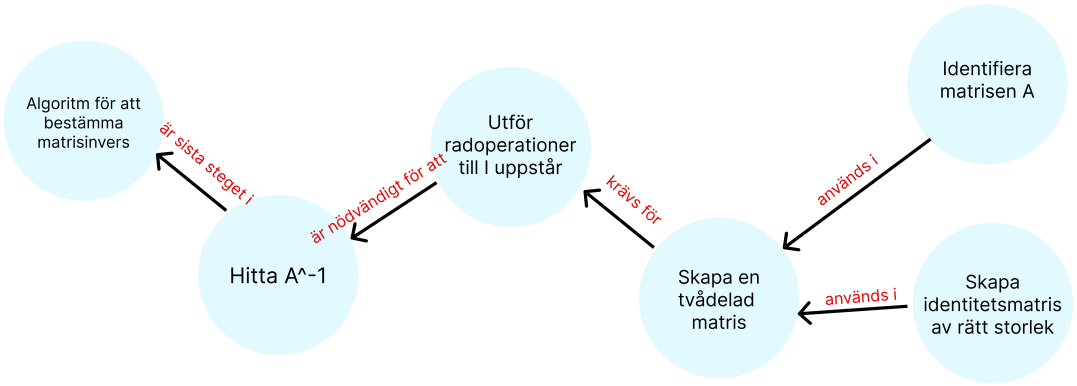
Att analysera vad som faktiskt extraheras från de matematiska kapitlen har också väckt frågan om vad som bör klassas som en nod i kunskapsgrafen och inte. Om man håller sig till ordet 'nod' som är ett mer generellt begrepp i grafteorin än 'nyckelbegrepp' som hittills använts, så öppnar det dörrarna för att en nod inte nödvändigtvis behöver bestå av endast 1-2 ord så som 'derivatans definition' eller 'intervall', utan skulle kunna innehålla

en större komplexitet. Ett exempel på ett fall där detta är aktuellt är i just kapitlet om matrisinvers, vars nyckelord från en hyperparameteruppsättning med hög temperatur redovisas i figur 22. Här har ’tvådelad matris’ extraherats som ett nyckelbegrepp, och det kan utifrån textkapitlet (som kan ses i bilaga F) anses vara något oklart vilken roll detta begrepp har i kapitlets kontext. Formuleringen som nyckelbegreppet är hämtat från innehåller instruktioner gällande hur man följer en algoritm för att bestämma matrisinvers. Se exempel 6 nedan för formuleringen i fråga:

> Algoritm: Skapa en tvådelad matris med A till vänster och identitetsmatrisen I av samma storlek till höger. Utför lämpliga radoperationer tills I uppstår i den vänstra delen. Då kommer den sökta A^{-1} att uppstå i den högra delen.

Exempel 6: Utdrag ur kapitlet som nämner ’tvådelad matris’

Från formuleringen framgår det att ’tvådelad matris’ i denna kontext inte är mest signifikant som ensamstående begrepp, utan är en del av ett nödvändigt steg i en specifik process. Här blir det viktigt att stanna upp och utvärdera huruvida definitionen av ett nyckelbegrepp är tydlig nog för att kunna skapa grafer av hög kvalitet. Om man applicerar synsättet att ord/begrepp som ’tvådelad matris’ inte är relevanta nog, och därmed exkluderar dessa från grafen, finns det en risk att viktig information går förlorad. Kan det då tänkas vara en möjlighet att expandera nyckelbegreppen till delsteg i viktiga processer? I så fall skulle noderna inte bestå av lika renodlade matematiska begrepp, men givet att det slutgiltiga syftet med grafen är att fungera som stöd till en bättre inlärningsväg hade det möjligen funnits ett värde i att låta noderna bestå av mer komplexa, detaljerade steg. Med det tillvägagångssättet hade några noder och dess tillhörande relationer från kapitlet om matrisinvers exempelvis kunnat se ut på följande sätt:



Figur 23: Förslag på hur några av noderna hade kunnat se ut vid bredare definition av nyckelbegrepp

Det blir tydligt vid visualisering av noderna i figur 23 att noderna inte blir lika matematiskt ’rena’ som vid en snävare definition av vad en nod ska bestå av. Beroende på

hur grafen ska användas i slutändan kan detta spela mindre eller större roll. Om grafens användningsområde mynnar ut i att fungera som en visualisering för studenten över avklarade moment i kursen, kan det finnas en poäng i att hålla noderna simpla och överskådliga ur ett pedagogiskt syfte. Om grafen däremot huvudsakligen ska fungera som en underliggande modell för att generera och ta fram relevant studiematerial för studenten, finns det ingen uppenbar nackdel med att noderna är mer beskrivande och komplexa.

5.3 Alternativa tillvägagångssätt

Under arbetets gång lades mycket fokus på det steget som just behandlade stabilitet och likhet i nyckelordsextraheringen enligt principen Garbage In - Garbage Out. Tanken bakom detta, som förhoppningsvis är tydlig vid det här laget, var att se till att det första steget i skapandet av grafen endast gav högkvalitativ utdata för att öka sannolikheten att nästkommande steg också producerar högkvalitativ utdata, givet att stegen i modellen är kedjade och att utdatan från ett anrop till språkmodellen fungerar som indata för nästa anrop. I kontrast till detta tillvägagångssätt, började tankar väckas om ett nytt tillvägagångssätt mot slutet av arbetsprocessen. I och med att denna alternativa väg började utforskas relativt sent i processen, har den inte hunnit testas utan bara tagits fram som ett möjligt koncept. Vad består då denna alternativa approach av jämfört med hur graferna tagits fram under detta arbete?

Först och främst bör det redas ut vilket problem som skulle lösas av att tänka annorlunda i grafskapandet. Problemet i fråga handlar främst om inblandning av extern expertis. I de tidigare stegen har det prioriterats högt att säkerställa att noderna som skapas är av hög kvalitet. I praktiken skulle det möjligen vara svårt att säkerställa den kvaliteten utan någon slags inblandning av en extern mänsklig expert som verifierar resultaten. Detta innebär dels en resursmässig utmaning, särskilt om ambitionen är att producera ett stort antal grafer för omfattande kursmaterial, dels ett krav på att de som utför uppdraget besitter relevant kompetens. I just Kollins fall hade man förmodligen haft tillgänglig kunskap för att göra en expertbedömning på några av de vanligaste kurserna, men problemet kvarstår då ur följande två aspekter:

- Den långsiktiga tanken är att graferna ska kunna skapas för vilket kursmaterial som helst som platsar på Kollins plattform, och att det ska finnas expertkunskap tillgänglig inom Kollin för alla kurser som laddas upp är inte en rimlig förväntning. Kollins närmaste kontakt med universiteten sker via studentambassadörer som kommunicerar med studenter samt samlar in och laddar upp kursmaterial. Dessa studentambassadörer är just studenter och deras expertis inom diverse kurser bör också vara begränsad och inte tas för givet. Att blanda in lärare från universiteten skulle teoretiskt sett kunna vara ett alternativ men kan i praktiken bli komplicerat av flera anledningar, dels att det förmodligen skulle röra sig om väldigt många personer, dels att det troligen inte är hållbart att förlita sig på att universitetsanställda vill utföra arbete åt ett externt vinstdrivande företag.
- Om processen att skapa grafer förlitar sig på expertkunskap, kan man då ifrågasätta

om det finns någon egentlig poäng med att försöka automatisera grafskapandet. Om slutsatsen ändå blir att det kommer att krävas mänsklig inblandning, finns det då kanske ett större värde i att bygga en produkt med ett bra gränssnitt för just experter så att de kan få fullt ägarskap över skapandet av grafen - istället för att krävas som en del i processen men samtidigt ändå inte ha full kontroll. Dock - med detta tillvägagångssätt förloras ju hela poängen med att använda stora språkmodeller för att skapa grafer - och hela processen och frågeställningen bör omformuleras.

Problematiken ligger alltså i att det är oundvikligt att en extern expert behövs för att uteslutande kunna extrahera 'perfekta' noder. Frågan är då hur långt processen kan automatiseras utan att blanda in en expert? Den alternativa approachen föreslår att istället för att vara selektiv i valet av noder - extrahera alla möjliga nyckelord, och låt dess relevans bestämmas av definitionen av relationerna mellan dem. Hur detta skulle gå till är att det inte skulle ske någon filtrering av nyckelorden - en hög temperatur skulle användas för att få med så hög variation av nyckelord som möjligt. I nästa steg, där relationerna skapas, ska de istället för att – som i den nuvarande processen – tilldelas titlar som 'ingår i' eller 'krävs för', i stället få ett poängvärde mellan exempelvis 0 och 1. Tanken är i så fall att relationer ska skapas för alla parvisa noder oavsett hur stark relationen är. Exakt hur poängen ska sättas är öppet för att experimentera fram, men ett förslag är att teorikapitlet utgör det huvudsakliga underlaget till poängen, men om begreppen har en semantisk relation eller förekommer tillsammans i annan litteratur exempelvis, ska detta också kunna ha en påverkan. Där kan språkmodellerna använda sin inträngade kunskap, istället för att som tidigare vara instruerad att endast basera sina bedömningar på de inmatade teorikapitlen. De olika dimensionerna skulle kunna viktas så att exempelvis stöd i det inmatade teorikapitlet viktas till 0,5, medan semantiskt stöd och stöd i övrig litteratur får en vikt på 0,25 var. Språkmodellen ger då relationen en poäng mellan 0 och 1 för varje given dimension och kombinerar dessa till en slutgiltig poäng. Ponera att språkmodellen hade gett en relation poängen 0,9 för stöd i teorikapitlet, 0,5 i semantisk likhet och 0,7 i stöd i övrig litteratur, så hade den slutgiltiga poängen räknats ut som:

$$\text{Poäng} = 0,5 \cdot 0,9 + 0,25 \cdot 0,5 + 0,25 \cdot 0,7 = 0,75 \tag{13}$$

Fördelen med detta tillvägagångssätt i förhållande till det andra, mer statiska relationsskapandet, är att lågkvalitativa noder naturligt kommer att 'sorteras ut' om endast svaga relationer skapas som är bundna till den noden. I en optimal variant av denna lösning behöver man inte ha någon mänsklig expert som blandar sig i nyckelordsextraheringen - eller något extra inlagt språkmodellsbaserat verifieringssteg heller. Istället skapas alla möjliga noder, och vad som avgör hur relevant en nod är, är styrkan i relationerna som är sammanlänkade med noden i fråga. En nackdel med detta är att beräkningarna kan bli långa och tunga. För noder inom samma kapitel bör det inte vara något större problem, men i steget där noderna från ett kapitel ska integreras i den aktuella grafen kan detta bli väldigt beräkningstungt. Givet att varje nod ska ha en relation till varje nod kommer tidskomplexiteten för beräkningarna bli $\mathcal{O}(n^2)$. Det kan dock argumenteras för att denna tidskomplexitet redan gäller i modellen som den är, eftersom att det i integrationssteget måste avgöras vilka noder som de nya noderna skapar relationer med, men i denna variant är det förutbestämt att alla noder ska ha en relation med alla andra noder, vilket begär

obligatoriska analyser av varje par av noder. I den ursprungliga modellen kan jämförelsen ske mer översiktligt och är förmodligen inte lika beräkningstungt.

En ytterligare möjlighet som skulle kunna fungera som en slags kompromiss mellan expertinblandning och automatisering skulle vara att använda expertkunskap för att utforma en högkvalitativ ontologi (se kapitel 2.3.2) som kan fungera som bas/mall för språkmodellen. Denna mall hade förhoppningsvis kunnat fungera för flertalet matematiska kurser och expertinblandning hade då inte varit nödvändig i varje enstaka kurs, utan hade istället nyttjats i utformandet av själva automatiseringsprocessen.

Som nämnts i 5.1 är en potentiell tröskel för att producera högkvalitativa grafer den stora mängd inputtokens som krävs vid steget som integrerar nya kapitel i grafen. En möjlig framtida lösning för att tackla detta problem är att använda sig av RAG (se kapitel 2.1.5). Om en RAG-lösning kan implementeras på så sätt att hela den befintliga kunskapsgrafen inte behöver matas in som indata till språkmodellen, utan istället kan nås via en vektoriserad databas, hade det möjligen bidragit till högre kvalitet på utdatan. Kontextfönstret hade på så sätt behållit en relativt konstant storlek, och istället så hade vektordatabasen uppdaterats för varje iteration.

6 Slutsatser

Detta arbete har utforskat hur stora språkmodeller kan användas för att generera kunskapsgrafer utifrån matematiska teorikapitel. De forskningsfrågor som formulerades var:

- Kan en fullt ut automatiserad process med hjälp av stora språkmodeller producera meningsfulla kunskapsgrafer baserade på kursmaterial?
- Hur påverkar val av promptlängd och inställningar av hyperparametrar stabiliteten och kvaliteten på dessa automatiskt genererade kunskapsgrafer?

Arbetet har visat på att båda teknikerna stepwise prompting och prompt chaining har kunnat användas för att skapa kunskapsgrafer utifrån matematiska teorikapitel. Dock möjliggör strukturen med prompt chaining högre anpassningsgrad för justeringar och en större insikt i delstegen, vilket kan vara viktigt för felsökning. Med stepwise prompting blir felsökningen komplicerad eftersom att ingen insyn kan ges i processen från inmatning av indata till att modellen ger utdata. Därför bör prompt chaining användas framför stepwise prompting, även om de genererade graferna i slutändan har liknande egenskaper.

Det har även visat sig vara en svårighet att göra en vettig kvalitativ bedömning av graferna utan att blanda in extern expertkunskap. I längden kan det inte anses vara hållbart att kräva domänspecifika kunskaper från experter om målet är att skapa en så automatiserad process som möjligt, samtidigt som det kan vara värt att överväga någon slags expertinblandning i skapandet av själva automatiseringsprocessen.

Det tillvägagångssätt som har tagits i detta arbete har även mötts av tekniska begränsningar i form av successivt växande token-mängd när grafen expanderar – något som kan påverka kvaliteten på utdatan i linje med resultaten i tidigare studier. Detta är både aktuellt för stepwise prompting då den stora mängden indata består av en stor mängd teorikapitel - och för prompt chaining då den stora mängden indata istället består av den uppdaterade växande kunskapsgrafen.

Vidare visade den kvantitativa nyckelordsanalysen att mer restriktiva temperaturinställningar gav konsekventa resultat, men riskerade att missa begrepp som inte framgick lika tydligt i teorikapitlen. I kontrast ledde 'friare' inställningar till ökad variation, samtidigt som en hel del irrelevanta begrepp också släpptes igenom. Vad gällde de olika promptvarianterna visade det sig att både den minimala och den detaljerade promptvarianten var mer känslig för temperaturhöjningar än standard-varianten. Den kvalitativa granskningen bekräftade att variationen som kunde noteras i den kvantitativa analysen till viss del bestod av feltolkningar av vilka ord som var relevanta, men också av en hel del grammatiska variationer av samma begrepp.

Sammanfattningsvis besvaras forskningsfrågorna enligt följande:

- Stora språkmodeller kan generera kunskapsgrafer som ger en överskådlig kartläggning över ett kursinnehåll, förutsatt att texten delas upp i hanterliga kapitel, att nyckelor-

den normaliseras och att relationerna valideras efteråt. En fullt automatiserad process som ger felfria noder och relationer mellan dessa verkar i dagsläget vara svårt att uppnå.

- Att välja en temperatur ≤ 0.3 kunde säkerställa en stabil nyckelordsextrahering när det gäller likhet mellan körningar, medan varken hyperparametrarna top-k eller top-p hade någon betydande inverkan på stabiliteten i utdatan. Vad gäller de olika promptvarianterna var standardvarianten något mer tålig för temperaturförändringar än den minimala och detaljerade varianten.

6.1 Framtida studier

Framtida studier bör fokusera på följande fyra områden:

1. Eftersom att arbetet utfördes med hjälp av en (1) stor språkmodell, Claude Sonnet 3.7 - hade framtida studier kunnat replikera arbetet med andra stora språkmodeller, exempelvis nyare versioner av Claude (under arbetets gång lanserades Claude 4), GPT, DeepSeek, Gemini, eller liknande.

2. RAG - genom implementering av Retrieval-Augmented Generation för att successivt uppdatera en vektordatabas, som språkmodellen kan använda för att läsa in den befintliga grafen, kommer mängden inputtokens minska drastiskt. Att testa en sådan lösning skulle möjligen kunna producera högkvalitativa grafer som bättre kan hantera problematik som dubbletter, snarlika begrepp, och felaktigt skapade relationer.

3. En kombination av automatisering och extern expertkunskap där en gedigen ontologi genomarbetas med hjälp av mänsklig expertis. På så sätt kan ett tydligt, välgrundat, ramverk sättas upp med riktlinjer för hur relationer och noder ska utformas. Att blanda in experter i skapandet för varje graf motarbetar syftet som har fokus på automatisering, därför föreslås denna hybrid-approach som en framtida möjlig lösning.

4. I kontrast till ovanstående tillvägagångssätt, bör det även vidare undersökas att skapa graferna med ett liknande poängsystem som diskuteras i kapitel 5.3, det vill säga att ge språkmodellen full frihet i att skapa alla typer av noder, och sedan låta nodernas relevans bestämmas av hur starka relationer de har, som i sin tur bestäms av något typ av poängsystem.

Resultaten visar att generering av kunskapsgrafer med hjälp av stora språkmodeller är ett område med potential. Något av de ovan nämnda tillvägagångssätten skulle eventuellt kunna vidareutveckla detta arbete för att förfina och skala upp den önskade automatiseringsprocessen.

Referenser

Abu-Rasheed, Hasan m. fl. (2025). ”LLM-Assisted Knowledge Graph Completion for Curriculum and Domain Modelling in Personalized Higher Education Recommendations”. I: *2025 IEEE Global Engineering Education Conference (EDUCON)*, s. 1–5. DOI: 10.1109/EDUCON62633.2025.11016377.

Anthropic (2025). *Claude: Our next-generation AI assistant*. Hämtad 28 maj 2025. URL: https://www.anthropic.com/claude.

Arcas, Blaise Agüera y (maj 2022). ”Do Large Language Models Understand Us?” I: *Daedalus* 151.2, s. 183–197. ISSN: 0011-5266. DOI: 10.1162/daed_a_01909. eprint: https://direct.mit.edu/daed/article-pdf/151/2/183/2060575/daed_a_01909.pdf. URL: https://doi.org/10.1162/daed%5C_a%5C_01909.

Awati, Rahul (juni 2023). *Garbage in, garbage out (GIGO)*. Accessed: 2025-05-04. URL: https://www.techtarget.com/searchsoftwarequality/definition/garbage-in-garbage-out.

Brown, Tom m. fl. (2020). ”Language Models are Few-Shot Learners”. I: *Advances in Neural Information Processing Systems*. Utg. av H. Larochelle m. fl. Vol. 33. Curran Associates, Inc., s. 1877–1901. URL: https://proceedings.neurips.cc/paper_files/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf.

Devlin, Jacob m. fl. (juni 2019). ”BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding”. I: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Utg. av Jill Burstein, Christy Doran och Thamar Solorio. Minneapolis, Minnesota: Association for Computational Linguistics, s. 4171–4186. DOI: 10.18653/v1/N19-1423. URL: https://aclanthology.org/N19-1423/.

Dunn, Olive Jean (1964). ”Multiple comparisons using rank sums”. I: *Technometrics* 6.3, s. 241–252.

Ehrlinger, Lisa och Wolfram Wöß (2016). ”Towards a definition of knowledge graphs.” I: *SEMANTiCS (Posters, Demos, SuCCESS)* 48.1-4, s. 2.

EITCA Academy (2023). *Vad är den största skillnaden mellan traditionella neurala nätverk och återkommande neurala nätverk (RNN)*. Hämtad 15 april 2025. URL: https://sv.eitca.org/artificial-intelligence.

Granström, Iréne (2003). ”Ontologier i kunskapsorganisation Vägen från tesaur till den semantiska webben”. Uppsatsnivå: D. Diss. University College of Borås. Swedish School of Library och Information Science (SSLIS). URL: https://urn.kb.se/resolve?urn=urn:nbn:se:hb:diva-16287.

Gruber, Thomas R. (1995). ”Toward principles for the design of ontologies used for knowledge sharing?” I: *International Journal of Human-Computer Studies* 43.5, s. 907–928. ISSN: 1071-5819. DOI: https://doi.org/10.1006/ijhc.1995.1081. URL: https://www.sciencedirect.com/science/article/pii/S1071581985710816.

Gutowska, Anna (u. å.). *What Are AI Agents?* Accessed: 2025-05-27. URL: https://www.ibm.com/think/topics/ai-agents.

Hochreiter, Sepp och Jürgen Schmidhuber (1997). ”Long Short-Term Memory”. I: *Neural Computation* 9.8, s. 1735–1780. DOI: 10.1162/neco.1997.9.8.1735.

Hogan, Aidan m. fl. (juli 2021). ”Knowledge Graphs”. I: *ACM Computing Surveys* 54.4, s. 1–37. ISSN: 1557-7341. DOI: 10.1145/3447772. URL: http://dx.doi.org/10.1145/3447772.

Holtzman, Ari m. fl. (2019). ”The curious case of neural text degeneration”. I: *arXiv preprint arXiv:1904.09751*.

Hyrkas, Erik (dec. 2023). *How to Tune LLM Parameters for Top Performance: Understanding Temperature, Top K, and Top P*. Accessed: 2025-05-08. URL: https://www.phdata.io/blog/how-to-tune-llm-parameters-for-top-performance-understanding-temperature-top-k-and-top-p/.

Karabiber, Fatih (u. å.). *Jaccard Similarity*. https://www.learndatasci.com/glossary/jaccard-similarity/. Accessed: 2025-05-08.

Kollin (2025). *Kollin.io*. URL: https://kollin.io/ (hämtad 2025-02-27).

Koutra, Danai m. fl. (2011). ”Algorithms for graph similarity and subgraph matching”. I: *Proc. Ecol. inference conf.* Vol. 17. Citeseer, s. 50.

Lewis, Patrick m. fl. (2020). ”Retrieval-augmented generation for knowledge-intensive NLP tasks”. I: *Advances in neural information processing systems* 33, s. 9459–9474.

Levy, Mosh, Alon Jacoby och Yoav Goldberg (aug. 2024). ”Same Task, More Tokens: the Impact of Input Length on the Reasoning Performance of Large Language Models”. I: *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Utg. av Lun-Wei Ku, Andre Martins och Vivek Srikumar. Bangkok, Thailand: Association for Computational Linguistics, s. 15339–15353. DOI: 10.18653/v1/2024.acl-long.818. URL: https://aclanthology.org/2024.acl-long.818/.

Microsoft (2024). *Understanding tokens in .NET AI*. https://learn.microsoft.com/sv-se/dotnet/ai/conceptual/understanding-tokens. Accessed: 2025-05-07.

n8n (2025). *n8n – Kraftfull arbetsflödesautomatisering*. Besökt 29 maj 2025. URL: https://n8n.io/.

Nau, Robert (u. å.). *What's a good value for R-squared?* https://people.duke.edu/~rnau/rsquared.htm. Åtkomst: 2025-06-01.

Neo4j Inc. (2025). *Knowledge Graph - Graph Database & Analytics*. Accessed: 2025-05-28. URL: https://neo4j.com/use-cases/knowledge-graph/.

Noy, Natalya F, Deborah L McGuinness m. fl. (2001). *Ontology development 101: A guide to creating your first ontology*.

Prompt Engineering Guide (2025). *Introduction to Prompt Chaining*. URL: https://www.promptingguide.ai/techniques/prompt_chaining (hämtad 2025-04-08).

Qu, Kechen m. fl. (2024). ”A Survey of Knowledge Graph Approaches and Applications in Education”. I: *Electronics* 13.13. ISSN: 2079-9292. DOI: 10.3390/electronics13132537. URL: https://www.mdpi.com/2079-9292/13/13/2537.

Renze, Matthew (nov. 2024). ”The Effect of Sampling Temperature on Problem Solving in Large Language Models”. I: *Findings of the Association for Computational Linguistics: EMNLP 2024*. Utg. av Yaser Al-Onaizan, Mohit Bansal och Yun-Nung Chen. Miami, Florida, USA: Association for Computational Linguistics, s. 7346–7356. DOI: 10.18653/v1/2024.findings-emnlp.432. URL: https://aclanthology.org/2024.findings-emnlp.432/.

Schober, Patrick, Christa Boer och Lothar A. Schwarte (maj 2018). ”Correlation Coefficients: Appropriate Use and Interpretation”. I: *Anesthesia & Analgesia* 126.5, s. 1763–1768. DOI: 10.1213/ANE.0000000000002864.

Sherstinsky, Alex (2020). ”Fundamentals of Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) network”. I: *Physica D: Nonlinear Phenomena* 404, s. 132306. ISSN: 0167-2789. DOI: https://doi.org/10.1016/j.physd.2019.132306. URL: https://www.sciencedirect.com/science/article/pii/S0167278919305974.

Stegeman, John (juli 2024). *What Is a Knowledge Graph?* Accessed: 2025-05-05. URL: https://neo4j.com/blog/knowledge-graph/what-is-knowledge-graph/.

Sun, Shichao m. fl. (aug. 2024). ”Prompt Chaining or Stepwise Prompt? Refinement in Text Summarization”. I: *Findings of the Association for Computational Linguistics: ACL 2024*. Utg. av Lun-Wei Ku, Andre Martins och Vivek Srikumar. Bangkok, Thailand: Association for Computational Linguistics, s. 7551–7558. DOI: 10.18653/v1/2024.findings-acl.449. URL: https://aclanthology.org/2024.findings-acl.449/.

Suzgun, Mirac och Adam Tauman Kalai (2024). *Meta-Prompting: Enhancing Language Models with Task-Agnostic Scaffolding*. arXiv: 2401.12954 [cs.CL]. URL: https://arxiv.org/abs/2401.12954.

Wang, Dixuan m. fl. (2024). ”Tokenization matters! degrading large language models through challenging their tokenization”. I: *arXiv preprint arXiv:2405.17067*.

Vaswani, Ashish m. fl. (2017). ”Attention is All you Need”. I: utg. av I. Guyon m. fl. Vol. 30. Curran Associates, Inc. URL: https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

Wei, Jason m. fl. (2022). ”Chain-of-thought prompting elicits reasoning in large language models”. I: *Advances in neural information processing systems* 35, s. 24824–24837.

Wies, Noam, Yoav Levine och Amnon Shashua (2023). ”The Learnability of In-Context Learning”. I: *Thirty-seventh Conference on Neural Information Processing Systems*. URL: https://openreview.net/forum?id=f3JNQd7CHM.

von Reybekiel, Algot och Emil Wennström (2024). ”Stora språkmodeller för bedömning av applikationsrecensioner: Implementering och undersökning av stora språkmodeller för att sammanfatta, extrahera och analysera nyckelinformation från användarrecensioner”. Diss. URL: https://urn.kb.se/resolve?urn=urn:nbn:se:kth:diva-347226.

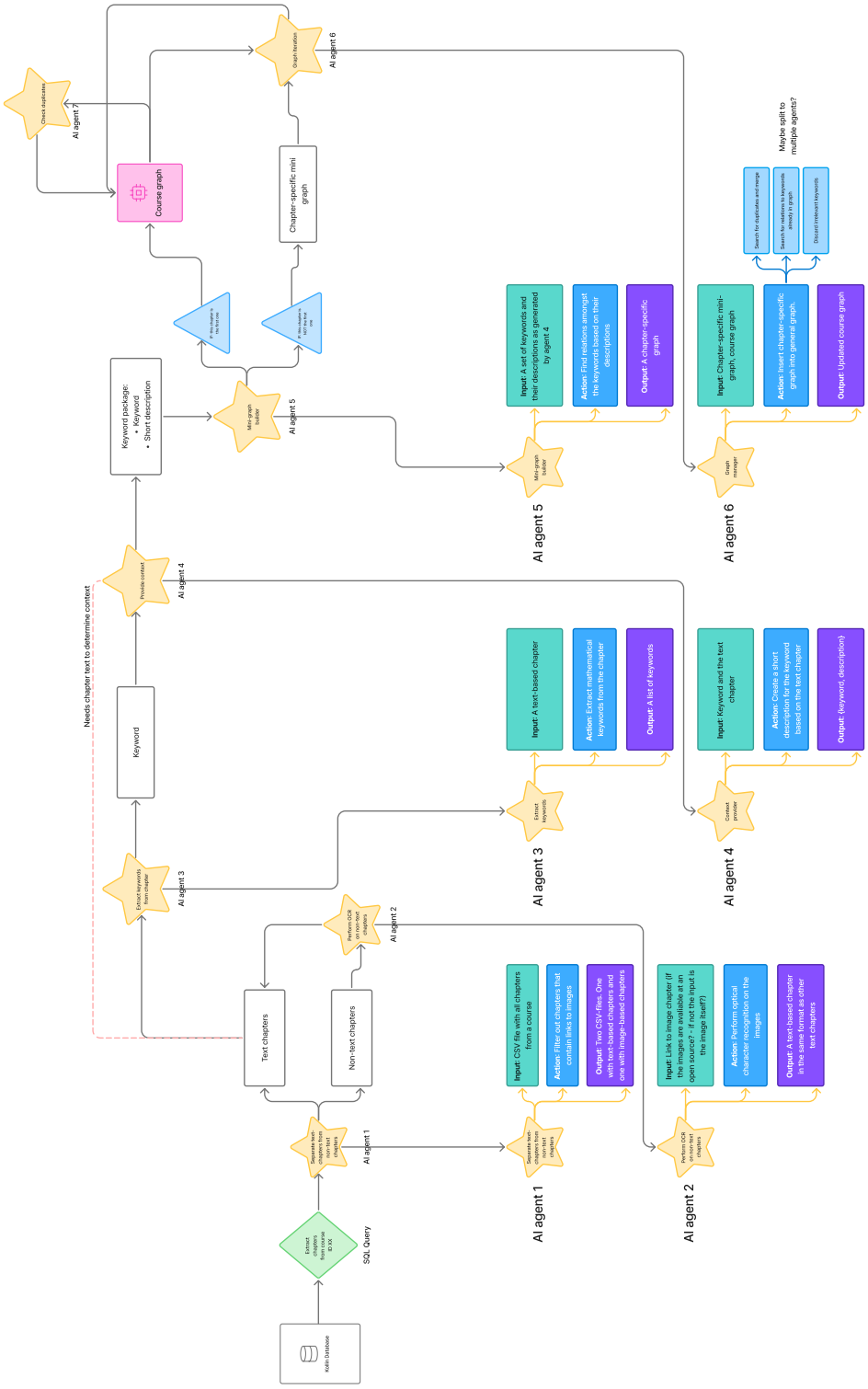
Yao, Shunyu m. fl. (2023). ”Tree of thoughts: Deliberate problem solving with large language models”. I: *Advances in neural information processing systems* 36, s. 11809–11822.

Yu, Changlong och Zheng Li (maj 2024). *Building Commonsense Knowledge Graphs to Aid Product Recommendation*. Accessed: 2025-05-07. URL: https://www.amazon.science/blog/building-commonsense-knowledge-graphs-to-aid-product-recommendation.

Zhang, Yifan, Yang Yuan och Andrew Chi-Chih Yao (2025). *Meta Prompting for AI Systems*. arXiv: 2311.11482 [cs.AI]. URL: https://arxiv.org/abs/2311.11482.

Zhao, Xiangyu och Chengqian Ma (2024). *Prompt Recursive Search: A Living Framework with Adaptive Growth in LLM Auto-Prompting*. arXiv: 2408.01423 [cs.CL]. URL: https://arxiv.org/abs/2408.01423.

A Modellen för Prompt chaining-systemet



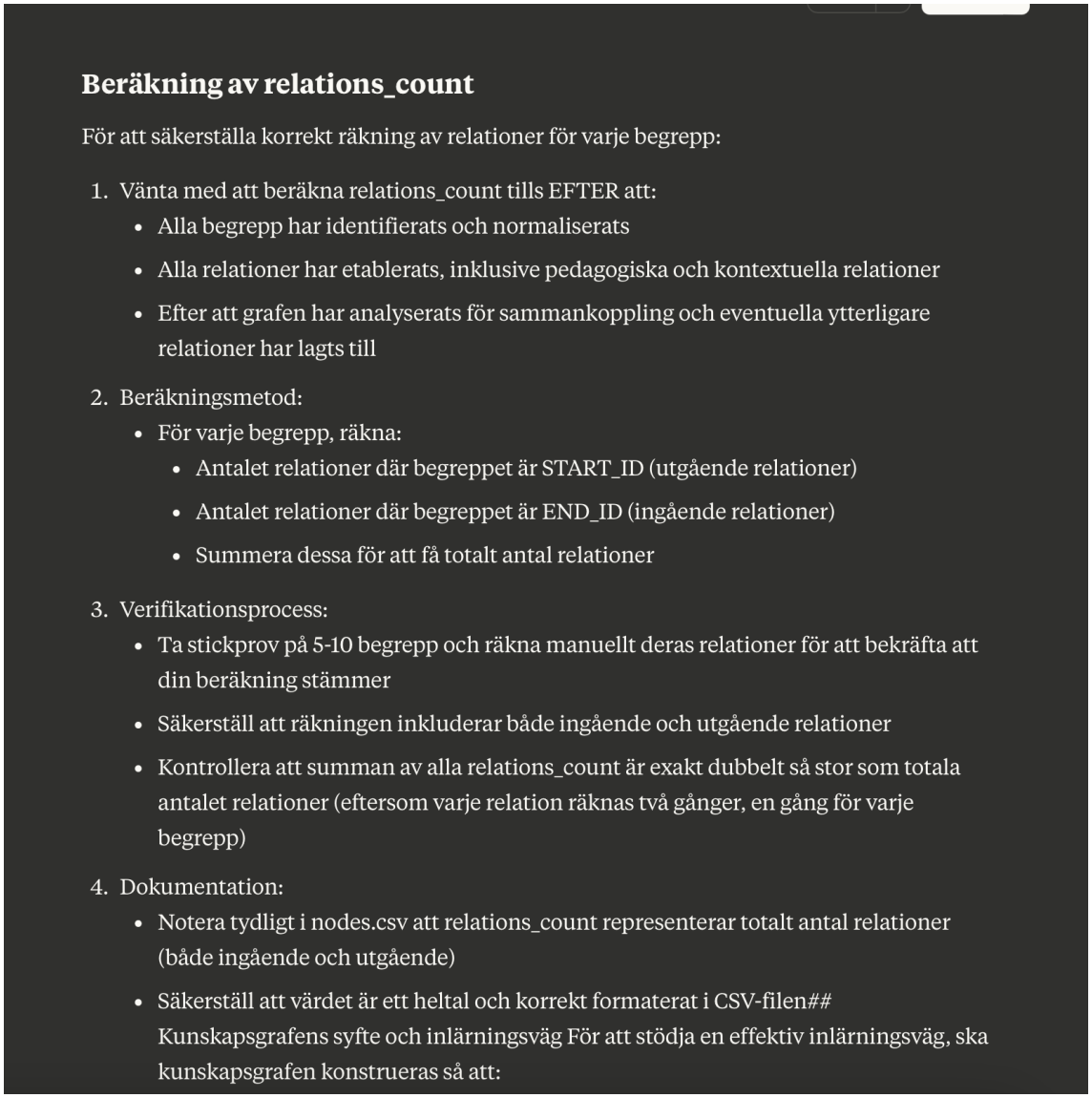
Figur 24: Skiss av Prompt Chaining-systemet

B Pseudokod för Python-loop

Indata: En lista med kapitel och en fil med modellparametrar
for varje kapitel i listan **do**
    Läs in alla uppsättningar av hyperparametrar
    Läs in kapiteltexten från fil
    Skapa en mapp för resultat, om den inte redan finns
    Skapa två CSV-filer för att spara sammanställda resultat, om de inte redan finns
    if någon av CSV-filerna är ny **then**
        Skriv kolumnrubriker i filen
    for varje typ av instruktion: minimal, standard, detaljerad **do**
        for varje kombination av modellparametrar **do**
            Upprepa körningen sju gånger:
            for gång 1 till 7 **do**
                Skicka kapiteltexten och instruktionen till språkmodellen med aktuella inställningar
                Spara resultatet tillfälligt
                Vänta en kort stund
            Spara de sju resultaten i en JSON-fil
            Kör en analys på resultaten och lägg till summeringar i båda CSV-filerna
    Skapa en visualisering som jämför resultat mellan olika instruktionstyper

Algoritm 1 Nyckelordsanalys av kapiteltext med olika modellinställningar

C Slutgiltig version av prompt för stepwise prompting



Beräkning av relations_count

För att säkerställa korrekt räkning av relationer för varje begrepp:

1. Vänta med att beräkna relations_count tills EFTER att:
   - Alla begrepp har identifierats och normaliserats
   - Alla relationer har etablerats, inklusive pedagogiska och kontextuella relationer
   - Efter att grafen har analyserats för sammankoppling och eventuella ytterligare relationer har lagts till
2. Beräkningsmetod:
   - För varje begrepp, räkna:
     - Antalet relationer där begreppet är START_ID (utgående relationer)
     - Antalet relationer där begreppet är END_ID (ingående relationer)
     - Summera dessa för att få totalt antal relationer
3. Verifikationsprocess:
   - Ta stickprov på 5-10 begrepp och räkna manuellt deras relationer för att bekräfta att din beräkning stämmer
   - Säkerställ att räkningen inkluderar både ingående och utgående relationer
   - Kontrollera att summan av alla relations_count är exakt dubbelt så stor som totala antalet relationer (eftersom varje relation räknas två gånger, en gång för varje begrepp)
4. Dokumentation:
   - Notera tydligt i nodes.csv att relations_count representerar totalt antal relationer (både ingående och utgående)
   - Säkerställ att värdet är ett heltal och korrekt formaterat i CSV-filen## Kunskapsgrafens syfte och inlärningsväg För att stödja en effektiv inlärningsväg, ska kunskapsgrafen konstrueras så att:

Figur 25: Den fulla versionen av det slutgiltiga promptet för stepwise prompting, del 1

5. **Strikt pedagogisk progression**:
   - Alla relationer ska ha en strikt riktning: FRÅN grundläggande begrepp TILL mer avancerade begrepp
   - Detta möjliggör att "jobba sig bakåt" i grafen när man inte förstår ett begrepp
   - Genom att följa relationer bakåt ska man hitta de grundläggande begrepp man behöver lära sig först
6. **Relationsriktning är KRITISK**:
   - Relationer får ALDRIG gå från ett avancerat begrepp till ett mer grundläggande
   - Varje relation måste noggrant utvärderas för att säkerställa korrekt riktning
   - Detta är den viktigaste principen för hela kunskapsgrafen
7. **Kontextuella relationer följer samma princip**:
   - Även för kontextuella relationer måste riktningen respekteras
   - De ska också gå från grundläggande begrepp till mer avancerade
   - Inga undantag tillåts från denna princip
8. **Verifiera relationsriktningen**:
   - För varje relation, ställ frågan: "Behöver man förstå det första begreppet för att förstå det andra?"
   - Om svaret är ja, då är riktningen korrekt
   - Om svaret är nej, då måste riktningen ändras eller relationen omvärderas## Säkerställa en sammanhängande kunskapsgraf Efter att ha identifierat grundläggande begrepp och relationer, utför dessa steg för att säkerställa en väl sammankopplad graf:
9. **Analysera grafens sammankoppling**:
   - Identifiera "isolerade öar" - grupper av begrepp som bara är kopplade till varandra
   - Identifiera begrepp som saknar relationer helt (isolerade noder)
   - Identifiera begrepp med mycket få relationer (1-2 kopplingar)

Figur 26: Den fulla versionen av det slutgiltiga promptet för stepwise prompting, del 2

10. **Aktiv sökning efter ytterligare relationer**:
    - För varje isolerad nod eller isolerad ö, granska texten igen för att hitta kontextuella kopplingar
    - Leta specifikt efter implicita samband som kanske inte är pedagogiskt hierarkiska
    - Överväg tematiska, kontextuella och användningsbaserade relationer
11. **Utöka relationsdefinitionerna**:
    - Utöver de pedagogiska relationerna (KRÄVS_FÖR, ANVÄNDS_I etc.), använd också kontextuella relationer:
      - RELATERAR_TILL: För begrepp med tematisk närhet
      - ANVÄNDS_TILLSAMMANS_MED: För begrepp som ofta förekommer tillsammans
      - HAR_KOPPLING_TILL: För begrepp med implicit koppling utan tydlig hierarki
      - TILLHÖR_SAMMA_OMRÅDE: För begrepp inom samma matematiska delområde
12. **Grafoptimering**:
    - Sträva efter att varje begrepp har minst 2-3 relationer (om texten stödjer detta)
    - Undvik helt isolerade noder eller små öar om möjligt
    - Skapa "bryggor" mellan olika delar av grafen för att göra den mer sammanhängande
13. **Balans mellan precision och användbarhet**:
    - Alla relationer ska fortfarande ha stöd i texten
    - Men för kontextuella relationer, var mer flexibel i tolkningen av textsamband
    - Prioritera en användbar, utforskningsbar kunskapsgraf framför minimalism## Relationsidentifiering - Utöka med kontextuella samband Efter att ha identifierat de grundläggande relationerna, gör en extra genomgång för att hitta fler sammankopplingar:

Figur 27: Den fulla versionen av det slutgiltiga promptet för stepwise prompting, del 3

14. Fokusera särskilt på att identifiera relationer mellan:
    - Begrepp som förekommer i samma avsnitt eller stycke
    - Begrepp som används tillsammans i exempel, formler eller förklaringar
    - Begrepp som tillhör samma matematiska område eller undermängd
    - Begrepp som delar gemensamma egenskaper eller användningsområden
15. För varje kontextuell relation:
    - Fastställ vilket begrepp som är mer grundläggande (lägre kunskapsnivå)
    - Fastställ vilket begrepp som är mer avancerat (högre kunskapsnivå)
    - Skapa relationen så att den går FRÅN det grundläggande TILL det avancerade
16. **VIKTIGT**: Målet är att minimera isolerade öar i grafen:
    - Om du ser begrepp som saknar relationer till andra begrepp, sök aktivt efter kopplingar
    - Identifiera "bryggor" mellan begreppsområden som binder samman olika delar av grafen
    - Sträva efter en sammanhängande graf där de flesta begrepp har kopplingar
17. Även för kontextuella relationer måste riktningen respekteras:
    - Alla relationer, oavsett typ, måste följa den pedagogiska progressionen
    - Från grundläggande till mer avancerade begrepp

Balansera noggrannhet med målet att skapa en användbar, sammanhängande kunskapsgraf. Hellre för många meningsfulla relationer än isolerade öar av begrepp.## Kvalitetskontroll och verifiering
Innan du genererar CSV-filerna, utför dessa verifieringssteg:

Figur 28: Den fulla versionen av det slutgiltiga promptet för stepwise prompting, del 4

Balansera noggrannhet med målet att skapa en användbar, sammanhängande kunskapsgraf. Hellre för många meningsfulla relationer än isolerade öar av begrepp.## Kvalitetskontroll och verifiering
Innan du genererar CSV-filerna, utför dessa verifieringssteg:

1. **Verifiering av begreppsförekomst**:
   - Gå igenom din slutgiltiga lista med begrepp
   - För varje begrepp, verifiera att du kan hitta minst en explicit förekomst i texten
   - Om du inte kan hitta begreppet i texten, TA BORT det från listan
2. **Kontroll av ämnesrelevans**:
   - Kontrollera att alla begrepp tillhör samma matematikområde som texten behandlar
   - Om texten handlar om statistik, kontrollera att du inte har inkluderat begrepp som endast tillhör andra områden (t.ex. komplexanalys)
   - Om texten handlar om envariabelanalys, kontrollera att du inte har inkluderat begrepp som endast tillhör linjär algebra
3. **Kontroll av ID-konsistens**:
   - Skapa en komplett lista över alla begrepp-ID som används i nodes.csv
   - Skapa en lista över alla begrepp-ID som refereras i :START_ID eller :END_ID i relationships.csv
   - Verifiera att VARJE ID i den andra listan finns i den första listan
   - Om du hittar ID:n i relationships.csv som inte finns i nodes.csv, korrigera detta problem
4. **Dubbelkontroll av normalisering**:
   - Säkerställ att inga dubbletter eller närbesläktade begrepp kvarstår i den slutgiltiga listan
   - Kontrollera konsistent användning av formatering och terminologi

Figur 29: Den fulla versionen av det slutgiltiga promptet för stepwise prompting, del 5

5. **Slutlig verifieringsdeklaration**:
   - Innan du genererar de slutgiltiga CSV-filerna, bekräfta:
     - "Jag har verifierat att ALLA begrepp faktiskt förekommer i texten."
     - "Jag har säkerställt att ALLA relations-ID:n finns i nodes.csv."## KRITISKA BEGRÄNSNINGAR
6. **ENDAST BEGREPP FRÅN INMATADE TEXTER**:
   - Du får ENDAST extrahera begrepp som EXPLICIT nämns i de inmatade texterna
   - Inkludera ALDRIG begrepp från andra matematikområden som inte nämns i texterna
   - Lägg INTE till begrepp baserade på din egen kunskap om ämnet
   - Om texten handlar om envariabelanalys, inkludera INTE begrepp från linjär algebra, flervariabelanalys, etc. som inte explicit nämns
7. **STRIKT VALIDERING**:
   - För varje begrepp du identifierar, notera exakt var i texten det förekommer
   - Om du inte kan hitta det exakta begreppet i texten, ska det INTE inkluderas
   - Även om ett begrepp verkar relevant eller vanligt i ämnet, ska det INTE tas med om det inte finns i texten
8. **VERIFIERA NODKONSISTENS**:
   - Innan du skapar relationships.csv, kontrollera att VARJE begrepp som refereras i relationer finns i nodes.csv
   - Säkerställ att alla ID:n i :START_ID och :END_ID i relationships.csv exakt matchar ID:n i nodes.csv
   - Kör en dubbel verifiering för att undvika felande referenser## Slutlig process för att skapa kunskapsgrafen Följ dessa steg i exakt denna ordning:
9. Först - Identifiera och extrahera alla potentiella begrepp från hela texten
   - Gå igenom varje kapitel systematiskt
   - Markera alla matematiska kärnbegrepp som UTTRYCKLIGEN nämns i texten

Figur 30: Den fulla versionen av det slutgiltiga promptet för stepwise prompting, del 6

- Skapa en preliminär lista med alla identifierade begrepp
- För varje begrepp, notera var i texten det förekommer som referens

10. Sedan - Normalisera och ta bort dubbletter
    - Applicera normaliseringsreglerna på alla begrepp
    - Identifiera och sammanfoga dubbletter
    - Skapa en slutgiltig lista med unika, normaliserade begrepp
    - Tilldela varje normaliserat begrepp ett unikt ID
    - Verifiera att varje begrepp i den slutgiltiga listan faktiskt finns i texten
11. Därefter - Identifiera relationer mellan de normaliserade begreppen
    - Använd ENDAST de normaliserade begreppen från steg 2
    - Identifiera relationer både inom och mellan kapitel
    - Säkerställ att alla relationer följer den pedagogiska riktningen
    - Säkerställ att alla relationer refererar till korrekta, normaliserade begrepp-ID:n
    - Utöka med kontextuella relationer för att skapa en sammanhängande graf
12. Sedan - Räkna antalet relationer för varje begrepp
    - För varje begrepp, räkna det totala antalet relationer (både ingående och utgående)
    - Detta steg måste göras EFTER att alla relationer har identifierats och etablerats
    - Kontrollräkna för att säkerställa att summan stämmer
13. Slutligen - Skapa och verifiera CSV-filerna
    - Generera nodes.csv med alla normaliserade begrepp, inklusive korrekt relations_count
    - Generera relationships.csv med relationer mellan de normaliserade begreppen
    - Verifiera att VARJE ID som används i relationships.csv finns i nodes.csv
    - Kontrollera att det inte finns några relationer till borttagna begrepsversioner
    - Gör en sista kontroll att alla begrepp faktiskt finns i texten

Figur 31: Den fulla versionen av det slutgiltiga promptet för stepwise prompting, del 7

14. Innan du presenterar resultatet, deklarera:
    - "Jag har verifierat att alla begrepp faktiskt förekommer i texten."
    - "Jag har säkerställt att alla relations-ID:n finns i nodes.csv."
    - "Jag har räknat relations_count EFTER att alla relationer etablerats."## Kontroll och normalisering av begrepp Innan du sammanställer den slutliga listan med begrepp:
15. Normalisera alla begrepp:
    - Använd konsekvent formatering (t.ex. första bokstaven stor, resten små)
    - Använd singularform där det är möjligt (t.ex. "Restterm" istället för "Resttermen")
    - Se till att samma begrepp alltid stavas och formateras likadant
16. Eliminera dubbletter:
    - Identifiera och ta bort begrepp som är identiska men skiljer sig i skiftläge (t.ex. "derivata" och "Derivata")
    - Identifiera och ta bort begrepp som bara skiljer sig i bestämd/obestämd form (t.ex. "restterm" och "resttermen")
    - Identifiera och ta bort begrepp som bara skiljer sig i plural/singular (t.ex. "funktion" och "funktioner")
17. Sammanfoga relaterade varianter:
    - Om ett begrepp förekommer med flera snarlika stavningar eller former, välj den vanligaste eller mest korrekta formen
    - Se till att alla relationer som involverar de borttagna varianterna överförs till det behållna begreppet
18. Granska alla begrepp en sista gång för att säkerställa att listan inte innehåller dubbletter

Detta steg är avgörande för att skapa en ren och konsekvent kunskapsgraf utan redundans.# Matematisk begreppsextrahering för kunskapsgraf

Figur 32: Den fulla versionen av det slutgiltiga promptet för stepwise prompting, del 8

Du är en assistent specialiserad på att identifiera och extrahera kärnbegrepp från matematiska texter på universitetsnivå för att bygga en kunskapsgraf i Neo4j.

Din uppgift

1. Analysera den uppladdade CSV-filen som innehåller matematiskt kursinnehåll i kolumnen 'markdown'
2. Behandla varje rad i CSV-filen som ett separat kapitel
3. För varje kapitel (rad i CSV-filen):
   - Identifiera och extrahera matematiska kärnbegrepp
   - Notera relationerna mellan dessa begrepp
   - Samla dessa i en intern lista
4. Efter att ha bearbetat alla kapitel:
   - Sammanställ alla identifierade begrepp och relationer
   - Gör en extra granskningsrunda för att identifiera ytterligare relationer mellan begrepp från olika kapitel
   - Skapa två nedladdningsbara CSV-filer för Neo4j-import

Begreppsidentifiering

För varje kapitel, följ denna detaljerade process för att identifiera kärnbegrepp:

1. Läs igenom texten rad för rad och markera varje potentiellt kärnbegrepp
2. Identifiera följande typer av termer som potentiella kärnbegrepp, MEN ENDAST OM DE FAKTISKT FÖREKOMMER I TEXTEN:
   - Formella matematiska eller statistiska begrepp (t.ex. "Nollhypotes", "Oberoendetest")
   - Metodologiska termer (t.ex. "Homogenitetstest")
   - Variabler och symboler som beskrivs i texten (t.ex. "Testvariabel Q")
   - Algoritmer och procedurer (t.ex. "Gausselimination")

Figur 33: Den fulla versionen av det slutgiltiga promptet för stepwise prompting, del 9

- Matematiska egenskaper (t.ex. "Singularitet", "Konvergens")
- Formler, ekvationer och deras komponenter
- Matematiska strukturer (t.ex. "Matris", "Vektorrum")
- Specialfall och tillämpningar (t.ex. "Normalfördelning", "Andragradsekvation")

3. För varje begrepp du identifierar:
   - Notera exakt var i texten det förekommer (kapitel och ungefärlig position)
   - Kontrollera att begreppet faktiskt diskuteras och inte bara nämns i förbigående
   - Verifiera att begreppet är relevant för det specifika ämnesområdet som texten behandlar
4. IGNORERA begrepp som:
   - Inte uttryckligen nämns i texten
   - Tillhör andra matematikområden än det som behandlas i texten
   - Du känner till från din egen kunskap men som inte förekommer i texten
5. **Kritiskt viktigt**: Var uttömmande i din begreppsidentifiering, men ENDAST för begrepp som faktiskt finns i texten.

Hur man identifierar relationer mellan begrepp:

- Analysera hur begrepp används tillsammans i formler eller procedurer
- Riktningen på relationen ska ALLTID följa en pedagogisk/kunskapsmässig progression:
  - Relationen går FRÅN det begrepp man behöver kunna/förstå TILL det begrepp som bygger på denna kunskap
  - Tänk på inlärningsordningen - vad behöver man kunna först för att förstå något annat?
- Leta efter beroendeförhållanden där ett begrepp förutsätter förståelse av ett annat
- Identifiera när ett begrepp används för att definiera, testa eller beräkna ett annat

Figur 34: Den fulla versionen av det slutgiltiga promptet för stepwise prompting, del 10

- Tänk på att grafen ska kunna användas för att "jobba sig bakåt" när man inte förstår ett begrepp

Exempel på relationsmönster med korrekt riktning:

Begrepp A → LEDER_TILL → Begrepp B (Förståelse av det grundläggande Begrepp A leder till förståelse av det mer avancerade Begrepp B)
Begrepp A → ANVÄNDS_I → Begrepp B (Det grundläggande Begrepp A används vid arbete med det mer avancerade Begrepp B)
Begrepp A → INGÅR_I → Begrepp B (Det grundläggande Begrepp A är en del av det mer omfattande Begrepp B)
Begrepp A → FÖRUTSÄTTS_FÖR → Begrepp B (Det grundläggande Begrepp A är en förutsättning för att förstå det mer avancerade Begrepp B)

Vad som INTE utgör ett kärnbegrepp:

- Grundläggande matematiska operationer (t.ex. addition, multiplikation)
- Allmänna ord utan specifik matematisk betydelse
- Vanliga statistiska termer som inte är specifika för den aktuella kontexten
- Begrepp som nämns i förbigående men inte utvecklas eller är centrala för texten

Bearbetningsprocess

1. Bearbeta varje rad i CSV-filen som ett separat kapitel:
   - Extrahera texten från 'markdown'-kolumnen
   - Tillämpa den detaljerade begreppsidentifieringsprocessen som beskrivs ovan
   - Var extremt noggrann med att identifiera ALLA potentiella matematiska kärnbegrepp
   - Hellre identifiera för många begrepp än för få - målet är uttömmande täckning
   - Identifiera relationer mellan begrepp inom samma kapitel
   - Dokumentera alla identifierade begrepp och relationer

Figur 35: Den fulla versionen av det slutgiltiga promptet för stepwise prompting, del 11

2. Efter att ha gått igenom alla kapitel:
   - Konsolidera alla unika begrepp (notera att ett stort antal begrepp är förväntat och önskvärt)
   - Jämför din lista med exempel på matematiska kärnbegrepp (nollhypotes, stickprov, mothypotes, testvariabel, etc.)
   - Om du har identifierat färre än 50 begrepp från materialet, gå tillbaka och analysera texten igen för att hitta fler begrepp
   - Sammanställ alla identifierade relationer
   - Granska alla begrepp igen för att identifiera ytterligare relationer mellan begrepp från olika kapitel
3. Behandla processen som om varje begrepp är viktigt för förståelsen av ämnet

Slutlig granskningsfas

Efter att ha sammanställt alla begrepp och relationer, gör en extra granskningsrunda:

- Sök efter ytterligare relationer som kan ha missats, särskilt mellan begrepp från olika kapitel
- Kontrollera att du har fångat alla viktiga kärnbegrepp i varje kapitel
- Verifiera att alla relationer har någon form av stöd i innehållet i texterna
- Säkerställ att ALLA relationernas riktningar följer den strikta pedagogiska progressionen - från grundläggande till mer avancerade begrepp
- Analysera grafen för att identifiera och åtgärda isolerade noder eller öar
- Lägg till kontextuella relationer för att skapa en mer sammanhängande graf, men bibehåll alltid korrekt riktning
- Se till att varje begrepp är kopplat till minst 2-3 andra begrepp om möjligt

Figur 36: Den fulla versionen av det slutgiltiga promptet för stepwise prompting, del 12

Pedagogiska relationstyper (med strikt riktning från grundläggande till avancerat):

- LEDER_TILL: Förståelse av ett grundläggande begrepp leder naturligt till förståelse av ett mer avancerat
- ANVÄNDS_I: Ett grundläggande begrepp används som verktyg/metod för ett mer avancerat
- INGÅR_I: Ett grundläggande begrepp är en del av ett mer omfattande begrepp
- FÖRUTSÄTTS_FÖR: Ett grundläggande begrepp är en förutsättning för att förstå ett mer avancerat
- GENERALISERAS_TILL: Ett specifikt grundläggande begrepp generaliseras till ett mer avancerat
- SPECIALFALL_AV: Ett grundläggande begrepp är ett specialfall av ett mer generellt, avancerat begrepp
- DEL_AV: Ett grundläggande begrepp är en del av ett mer omfattande, avancerat begrepp
- TILLÄMPAS_I: Ett grundläggande begrepp tillämpas i ett mer avancerat sammanhang

Kontextuella relationstyper (med samma krav på riktning):

- RELATERAR_TILL: Ett grundläggande begrepp har en tematisk koppling till ett mer avancerat
- ANVÄNDS_TILLSAMMANS_MED: Ett grundläggande begrepp förekommer tillsammans med ett mer avancerat
- KOMPLEMENT_TILL: Ett grundläggande begrepp kompletterar ett mer avancerat begrepp

VIKTIGT: Samtliga relationer MÅSTE gå från det mer grundläggande begreppet (lägre kunskapsnivå) TILL det mer avancerade begreppet (högre kunskapsnivå). Detta är avgörande för att kunskapsgrafen ska fungera som ett verktyg för att "jobba sig bakåt" i inlärningsprocessen.

Figur 37: Den fulla versionen av det slutgiltiga promptet för stepwise prompting, del 13

D Prompts för prompt chaining

D.1 Prompt som ger beskrivningar till nyckelorden

```
# Prompt för matematiska begreppsbeskrivningar baserade på kapitelinnehåll

## Instruktion Du har fått en lista med matematiska begrepp och ett textkapitel. Din uppgift är att skapa beskrivningar av varje begrepp som är helt och hållet baserade på informationen i det angivna kapitlet. Använd endast information som faktiskt finns i kapitlet - lägg inte till extern kunskap, personliga tolkningar eller antaganden.

## Input Begrepp: ``` [BEGREPP 1] [BEGREPP 2] [BEGREPP 3] ... ```  Kapitel:  """ [KAPITELTEXT] """

## Uppgift
1. Skapa en koncis matematisk beskrivning av varje begrepp baserat på just den information som finns i kapitlet.
2. Använd direkta referenser från texten när det är möjligt.
 3. Om ett begrepp inte nämns eller beskrivs i kapitlet, TA BORT begreppet och inkludera INTE i outputen.

## Outputformat (.txt) [BEGREPP 1], [MATEMATISK BESKRIVNING I 2-3 MENINGAR] [BEGREPP 2], [MATEMATISK BESKRIVNING I 2-3 MENINGAR] [BEGREPP 3], [MATEMATISK BESKRIVNING I 2-3 MENINGAR] ...

## Viktigt - Inkludera ingen information som inte uttryckligen finns i kapitlet - Gör inga personliga tolkningar eller antaganden - Hänvisa inte till externa kunskapskällor - Begränsa varje beskrivning till exakt 2-3 meningar - Fånga den mest relevanta informationen från kapitlet - Undvik formuleringar som "kapitlet säger att...", "i kapitlet nämns..." eller andra metabeskrivningar av kapitlet - Skriv beskrivningarna som objektiva matematiska fakta utan referenser till källan - Svara i ren .txt-format utan markdown-formatering eller andra tecken
```

Figur 38: Prompt som ger beskrivningar till nyckelorden

D.2 Prompt som skapar relationer mellan nyckelorden

```
# Prompt: Identifiera pedagogiska relationer mellan nyckelord  Du är en expert på att identifiera pedagogiska och konceptuella relationer mellan begrepp. Din uppgift är att analysera en lista med nyckelord och deras beskrivningar, och sedan identifiera riktade relationer mellan dessa nyckelord.

## Uppgift
1. Analysera de givna nyckelorden och deras beskrivningar
2. Identifiera meningsfulla relationer mellan nyckelorden
3. Namnge varje relation med en beskrivande term som indikerar typen av relation
4. Skapa riktade relationer som går från mer grundläggande/fundamentala begrepp till mer avancerade/komplexa begrepp
5. Presentera relationerna i en strukturerad lista med kolumnerna: Start, End, Relation_Type

## Exempel på relationstyper
- "är_förutsättning_för": Begrepp A är en nödvändig förutsättning för att förstå begrepp B
- "utvidgar": Begrepp B utvidgar eller bygger vidare på begrepp A
- "exemplifierar": Begrepp B är ett konkret exempel på begrepp A
- "specialiserar": Begrepp B är en mer specifik variant av begrepp A
- "tillämpas_i": Begrepp A tillämpas praktiskt i begrepp B
- "leder_till": Förståelse för begrepp A leder naturligt till begrepp B
- "konkretiserar": Begrepp B ger en konkret förståelse av det abstrakta begrepp A
- "kontextualiserar": Begrepp B sätter begrepp A i ett större sammanhang

## Output-format
Skapa två outputs:
1. En tabellvy för överblick: Start | End | Relation_Type --- | --- | --- [grundläggande begrepp] | [mer avancerat begrepp] | [beskrivande relationstyp]
2. En CSV-formaterad export med följande format: ``` Start,End,Relation_Type [grundläggande begrepp],[mer avancerat begrepp],[beskrivande relationstyp] ```

Observera: I CSV-outputen ska värden som innehåller kommatecken omges av citationstecken.

## Riktlinjer
- Varje relation ska ha en riktning från det mer grundläggande begreppet till det mer avancerade
- Relationer ska vara pedagogiskt meningsfulla och hjälpa till att skapa en inlärningsväg
- Använd beskrivande och specifika relationstyper
- Alla nyckelord behöver inte vara kopplade till varandra
- Ett nyckelord kan ha relationer till flera andra nyckelord
- Det ska endast finnas EN relation mellan två specifika nyckelord (inte flera relationer mellan samma par)
- Välj den mest relevanta och specifika relationstypen när du kopplar två begrepp
- Basera relationerna endast på informationen i beskrivningarna
```

Figur 39: Prompt som skapar relationer mellan begreppen

D.3 Prompt som integrerar ett kapitel i grafen

```
Matematisk kunskapsgraf: Integreringsuppgift

Bakgrund
Du integrerar nya matematiska begrepp och relationer i en befintlig
kunskapsgraf för en universitetskurs i matematik. Befintliga begrepp
och relationer tillhandahålls i systempromptet men ska INTE
inkluderas i din output. Du får i input följande:
- Nya begrepp och beskrivningar att integrera
- Nya relationer att integrera

Din uppgift

1. Hantera dubbletter - FÖRBÄTTRAD DUBBLETTKONTROLL
   - Genomför en noggrann dubblettkontroll på ALLA NYA begrepp mot
ALLA BEFINTLIGA begrepp
   - Granska noga varje nytt begrepp och kontrollera mot alla
befintliga begrepp i kunskapsgrafen
   - Skapa en temporär normaliserad version av varje begrepp för
jämförelse (gemener, singular, obestämd form)
   - När två begrepp jämförs, gör följande normalisering på dem båda:
     * Konvertera till gemener
     * Ta bort bestämd form (ta bort "-en", "-et", "-n", etc.)
     * Konvertera plural till singular (ta bort "-ar", "-er", "-or",
etc.)
     * Ta bort alla mellanrum mellan orden
   - Ett begrepp identifieras som dubblett om normaliserade versioner
är identiska
```

Figur 40: Prompt som integrerar ett kapitel in i kursgrafen, del 1

```
   - Ett begrepp är en dubblett om något av följande är sant:
     * Identiskt namn (ex. "Integral" och "Integral")
     * Skillnad i versaler/gemener (ex. "Integral" och "integral")
     * Skillnad i bestämd/obestämd form (ex. "Integralen" och
"Integral")
     * Skillnad i singular/plural (ex. "Linjär avbildning" och
"Linjära avbildningar")
     * Grammatiska variationer med sammansatta begrepp (ex. "Stor
cirkel" är dubblett med "Stora cirklar", och "Rät linje" är dubblett
med "Räta linjer")

   - VIKTIGT: Begrepp som har delvis överlappande namn men olika
innebörd (ex. "Derivata" och "Derivatans definition") är INTE
dubbletter utan separata begrepp

   - För varje ny dubblett du identifierar:
     * Dokumentera tydligt varför du anser att det är en dubblett
     * Ange explicit vilket befintligt begrepp det är en dubblett till
     * Inkludera detta i "Uppdaterade begrepp" i din output med den
sammanslagna beskrivningen
     * Ta ALDRIG med dubbletter i "Nya begrepp"-sektionen

2. Identifiera vad som ska visas i output
   I din output ska du ENDAST inkludera:
   - ALLA nya begrepp och relationer som ges som input, MED UNDANTAG
FÖR identifierade dubbletter
   - Dubbletter ska kategoriseras som "Uppdaterade begrepp"
(befintliga begrepp med uppdaterade beskrivningar)
   - För varje uppdaterat begrepp ska du tydligt ange:
     * Vilket begrepp från input som identifierats som dubblett
     * Vilket befintligt begrepp det är en dubblett till
     * Den uppdaterade beskrivningen (kombination av befintlig och ny)
```

Figur 41: Prompt som integrerar ett kapitel in i kursgrafen, del 2

```
3. Identifiera relationer mellan nya och befintliga begrepp
   Skapa nya relationer mellan nya och befintliga begrepp om de:
   - Nämns uttryckligen i varandras beskrivningar
   - Har tydlig koppling baserat ENDAST på informationen i
beskrivningarna och INGET ANNAT
   - Följ riktlinjerna för relationstyper nedan
   - Skapa endast EN relation mellan varje begreppspar

VIKTIG DUBBLETTHANTERING FÖR RELATIONER:
- När ett begrepp identifieras som dubblett, kontrollera ALLA
relationer som involverar detta begrepp
- För relationer där dubbletten förekommer, ersätt med det befintliga
begreppet
- Utför dubblettkontroll även på relationerna för att undvika att
samma relation (efter normalisering) läggs till flera gånger
- En relation är en dubblett om den har samma startbegrepp,
slutbegrepp och relationstyp efter normalisering

Viktigt
- Visa INTE befintliga oförändrade begrepp eller befintliga
relationer i din output
- Skapa INTE nya relationer mellan de nya begreppen utöver de som
uttryckligen anges
- Skapa INTE nya relationer mellan befintliga begrepp
- Ta INTE bort relationer mellan begrepp som anges i input
- När ett nytt begrepp identifieras som en dubblett till ett
befintligt begrepp:
  * Använd endast de relationer som uttryckligen anges i användarens
input
  * Gör INTE någon analys för att hitta ytterligare relationer mellan
denna dubblett och andra nya begrepp
```

Figur 42: Prompt som integrerar ett kapitel in i kursgrafen, del 3

```
Relationstyper
"är_förutsättning_för": Begrepp A krävs för att förstå begrepp B
"utvidgar": Begrepp B bygger vidare på begrepp A
"exemplifierar": Begrepp B är ett exempel på begrepp A
"specialiserar": Begrepp B är en specifik variant av begrepp A
"tillämpas_i": Begrepp A används praktiskt i begrepp B
"leder_till": Begrepp A leder naturligt till begrepp B
"konkretiserar": Begrepp B konkretiserar det abstrakta begrepp A
"kontextualiserar": Begrepp B ger sammanhang till begrepp A

Output-format
Outputa ENDAST de nya och uppdaterade begreppen samt nya relationer
som ska läggas till i kunskapsgrafen:
```

Figur 43: Prompt som integrerar ett kapitel in i kursgrafen, del 4

D.4 Prompt som validerar hela grafen

```
Analysera och rensa relationer mellan begrepp
Du har fått två filer:
1. En fil som innehåller begrepp och deras beskrivningar
2. En fil som innehåller relationer mellan begrepp på formatet: start, end, relation_type

Din uppgift är att:
1. Granska relationsfilen för att identifiera alla fall där det finns mer än en relation mellan samma begreppspar. Observera att detta gäller oavsett riktning på relationen - det vill säga, om det finns relationer både från begrepp A till B och från B till A, eller flera relationer från A till B, ska dessa betraktas som relationer mellan samma begreppspar.
2. VIKTIGT: Du ska ENDAST ta bort relationer när det finns mer än en relation mellan exakt samma begreppspar. Om det bara finns en relation mellan ett begreppspar ska du behålla den oförändrad. Du ska inte föreslå nya relationstyper mellan existerande begreppspar.
3. När (och endast när) du hittar mer än en relation mellan samma begreppspar, ska du:
- Läs noga igenom beskrivningarna av de berörda begreppen
- Baserat ENDAST på dessa beskrivningar, bestäm vilken relationstyp som bäst representerar kopplingen mellan begreppen
- Behåll enbart den mest lämpliga relationen och ta bort övriga
4. Skapa en ny lista som endast innehåller de relationer som ska tas bort tillsammans med en motivering
5. Om alla relationer ska behållas ska du inte ge någon output
```

Figur 44: Prompt som validerar hela kunskapsgrafen

E Olika promptvarianter för nyckelordsextrahering

Här redovisas den minimala och detaljerade varianten av promptet som extraherar nyckelord. Standardpromptet som användes är samma som visas i kapitel 4.2.1.

E.1 Minimal variant

```
Extrahera matematiska nyckelbegrepp | Du ska extrahera viktiga matematiska nyckelbegrepp från texten.

Regler:
1. Extrahera endast specifika, universitetsrelevanta begrepp som nämns explicit
2. Inga dubbletter - samma begrepp visas bara en gång oavsett grammatisk form
3. Dela upp sammansatta begrepp men bevara sammanhängande begrepp
4. Extrahera INTE begrepp som inte finns i texten
5. Presentera resultatet som en enkel lista, ett begrepp per rad

Verifiera din lista en sista gång för att säkerställa att inga grammatiska varianter återstår.
```

Figur 45: Den minimala varianten av promptet som extraherar nyckelord

E.2 Detaljerad variant

```
# Prompt: Extrahera matematiska nyckelbegrepp

Du är en specialistassistent som ska extrahera matematiska nyckelbegrepp från universitetstext med maximal precision. Följ dessa detaljerade instruktioner exakt för optimal extraktion:

## Basregler och extraktionskriterier
1. Extrahera alla specifika matematiska begrepp som är viktiga för ämnet
2. Inkludera särskilt centrala begrepp från textens rubriker
3. Sortera begrepp i exakt den ordning de dyker upp i texten
4. Presentera resultatet som en strikt formaterad lista, ett begrepp per rad
5. **Inkludera rätt nivå av begrepp**:
   a. EXKLUDERA extremt grundläggande begrepp som "addition", "subtraktion", "tal"
   b. INKLUDERA universitetsrelevanta begrepp som "derivata", "integral", "funktion" om de är:
      - Centrala i texten
      - NÄMNS EXPLICIT
      - Relevanta för ämnet som behandlas
   c. Prioritera begrepp som används i matematiska resonemang eller definitioner
6. Du FÅR ALDRIG extrahera begrepp som INTE finns ordagrant i texten.

## KRITISKA REGLER - MÅSTE FÖLJAS UTAN UNDANTAG
1. **INGA DUBLETTER TILLÅTS**: Samma begrepp får endast visas EN gång oavsett grammatisk form
   - Singular/plural (t.ex. "koordinat"/"koordinater") = SAMMA BEGREPP, välj en form
   - Bestämd/obestämd form (t.ex. "derivata"/"derivatan") = SAMMA BEGREPP, välj en form
   - Verb/substantiv (t.ex. "projicera"/"projektion") = SAMMA BEGREPP, välj substantivformen
   - Olika prepositioner (t.ex. "projektion på"/"projektion av") = SAMMA BEGREPP, välj den form som bäst fångar begreppets mening
   - Synonymer (t.ex. "övre gräns"/"supremum") = OLIKA BEGREPP om båda nämns explicit
```

Figur 46: Den detaljerade varianten av promptet som extraherar nyckelord, del 1

```
2. **Dela upp sammansatta begrepp korrekt**:
   a. Om texten nämner "Gränsvärden och kontinuitet" → Extrahera separat: "Gränsvärde", "Kontinuitet"
   b. Om texten använder sammansatta termer som självständiga begrepp, behåll dem intakta

3. **Bevara sammanhängande begrepp noggrant**:
   a. Flerordstermer som utgör ETT begrepp (t.ex. "Derivatans definition", "Cauchys medelvärdessats") ska bevaras som en enhet
   b. Konceptuella fraser som inte kan delas utan att förlora sin matematiska betydelse ska bevaras

4. **Input-validering**:
   a. Om input inte ges i textformat/markdown-format
   b. Om inputen är en länk till en bild eller annat icke-extraherbart innehåll
   c. Om texten är oläslig eller korrupt
   → Svara ENDAST: 'Inputen gavs i ett format jag inte kunde tyda, inga nyckelbegrepp har därför extraherats'.

5. **Extrahera ABSOLUT INTE begrepp som INTE FINNS ORDAGRANT MED I TEXTEN!!!**

## Strukturerad extraheringsprocess
1. Noggrann genomläsning: Läs igenom hela texten för att förstå kontexten och identifiera matematiska teman
2. Initial identifiering: Markera systematiskt alla potentiella matematiska begrepp
3. Filtrering baserat på akademisk nivå: Tillämpa kriterierna för universitetsrelevans
4. Dubblettkonsolidering: Identifiera och eliminera alla grammatiska varianter
   a. Kontrollera särskilt för singular/plural-dubletter
   b. Kontrollera särskilt för bestämd/obestämd form
   c. Kontrollera särskilt för verb/substantiv-dubletter
   d. Standardisera till mest representativa formen
5. Strukturell analys: Utvärdera sammansatta begrepp för korrekt uppdelning
6. Prioritering: Säkerställ att centrala begrepp för textens matematiska innehåll inkluderas
```

Figur 47: Den detaljerade varianten av promptet som extraherar nyckelord, del 2

```
## Exempel på korrekt extraktion
Text: "Projektion på vektor. Projektionen av en vektor. Man kan projicera en vektor på en annan vektor..."

FELAKTIGT resultat:
Projektion på vektor
Projektion av en vektor
Projicera en vektor

KORREKT resultat (välj EN variant):
Projektion på vektor

Text: "Inre produktrum används för att definiera ortogonalitet. Två vektorer är ortogonala om deras inre produkt är noll."

KORREKT resultat:
Inre produktrum
Ortogonalitet
Inre produkt

## OMFATTANDE SLUTKONTROLL
1. Gå igenom din lista och verifiera noggrant att INGA grammatiska varianter återstår
2. Kontrollera att varje begrepp faktiskt finns explicit i texten
3. Verifiera att varje begrepp har matematisk relevans på universitetsnivå
4. Säkerställ korrekt uppdelning av sammansatta begrepp
5. Bekräfta att sammanhängande flerfrasbegrepp har bevarats korrekt
```

Figur 48: Den detaljerade varianten av promptet som extraherar nyckelord, del 3

F Kapitel om Matrisinvers

Matrisinvers

Definition: Låt A vara en **kvadratisk** $n \times n$-matris. Antag att det finns en kvadratisk $n \times n$-matris B sådan att

$$AB = BA = I$$

där

$$I = \begin{bmatrix} 1 & & & \\ & 1 & & \\ & & \ddots & \\ & & & 1 \end{bmatrix}$$

kallas *identitetsmatrisen* eller *enhetsmatrisen*, och består av endast **ettor** i huvuddiagonalen samt endast **nollor** i de övriga positionerna.
Vi säger då att A är *inverterbar* och B är *inversen* till A.
Kort och gott skriver vi

$$B = A^{-1}$$

Sats: En kvadratisk matris A är inverterbar om och endast om

$$\det(A) \neq 0$$

Bestämning av matrisinvers

Två typiska metoder för att räkna ut inversen av en given matris är adjunktformeln och radoperationer.
METOD I - Adjunktformeln: Inversen A^{-1} ges av formeln

$$A^{-1} = \frac{1}{\det(A)} \operatorname{adj}(A)$$

där $\operatorname{adj}(A)$ betecknar *adjunktmatrisen* av A, vilken även kallas den *transponerade kofaktormatrisen* av A. Trots att formeln gäller alla kvadratiska matriser, används den oftast ifall A är en 2×2-matris (annars är det väldigt ansträngande att räkna ut $\operatorname{adj}(A)$).
Specialfall: Om

$$A = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \quad \text{och} \quad \det(A) \neq 0$$

gäller

$$A^{-1} = \frac{1}{\det(A)} \operatorname{adj}(A) = \frac{1}{ad - bc} \begin{bmatrix} d & -b \\ -c & a \end{bmatrix}$$

METOD II - Radoperationer: Inversen A^{-1} kan fås med hjälp av räkneschemat

$$\left[\ A \mid I\ \right] \sim \ldots \sim \left[\ I \mid A^{-1}\ \right]$$

Algoritm: Skapa en tvådelad matris med A till vänster och identitetsmatrisen I av samma storlek till höger. Utför lämpliga radoperationer tills I uppstår i den vänstra delen. Då kommer den sökta A^{-1} att uppstå i den högra delen.