



Högskoleingenjörsutbildning i Datateknik

Grundnivå, 15hp

# Informationsextrahering från långa PDF-dokument för effektivt frågebesvarande

En jämförande studie av system för informationsutvinning och frågebesvarande över finansiella dokument med stora språkmodeller

**MAGNUS HEN**

**MARCUS VON SCHANTZ**



# **Informationsextrahering från långa PDF-dokument för effektivt frågebesvarande**

En jämförande studie av system för informationsutvinning och frågebesvarande över finansiella dokument med stora språkmodeller

## **Information Extraction from Long PDF Documents for Efficient Question Answering**

A Comparative Study of Information Extraction and Question Answering Systems for Financial Documents Using Large Language Models

Magnus Hen  
Marcus von Schantz

Examensarbete inom datateknik  
Grundnivå, 15 hp  
Handledare på KTH: Jonas Willén  
Examinator: Anders Lindström  
TRITA-CBH-GRU-2026:138

KTH  
Skolan för kemi, bioteknologi och hälsa  
141 52 Huddinge, Sverige



## Sammanfattning

Denna rapport undersöker hur stora språkmodeller kan användas för att automatisera informationsextraktion ur finansiella PDF-dokument. Finansiella dokument är ofta långa och komplexa, vilket gör manuell extrahering tidskrävande och svår att skala. Syftet är att ta fram och utvärdera ett sammanfattningsbaserat system samt jämföra dess prestanda mot ett Retrieval-Augmented Generation-system (RAG) för frågebesvarande. Det utvecklade sammanfattningsbaserade systemet består av fyra steg: textextrahering, segmentering, frågefokuserad och hierarkisk sammanfattning, och till sist frågebesvarande. Parametrarna som undersöks i arbetet är två PDF-läsare, tre segmentstorlekar samt två sammanfattningslängder. Utvärderingen genomfördes med måtten Exact Match och F1-värde. Den valda konfigurationen för det utvecklade systemet uppnådde ett genomsnittligt Exact Match-värde på 0,81 och genomsnittligt F1-värde på 0,85 över tre körningar. Den testade RAG-konfigurationen uppnådde motsvarande värden på 0,76 respektive 0,81. Resultaten indikerar att det utvecklade systemet kan utgöra ett relevant alternativ till retrieval-baserade strategier för frågebesvarande över finansiella dokument.

### Nyckelord

Stora språkmodeller, Retrieval-Augmented Generation, frågebesvarande, frågefokuserad sammanfattning, hierarkisk sammanfattning, informationsutvinning, PDF-extrahering, prompt engineering, Exact Match, F1-värde.



## **Abstract**

This study investigates how large language models can be used to automate information extraction from financial PDF-documents. Financial documents are often long and complex, which makes manual extraction time-consuming and difficult to scale. The purpose of this study is to develop and evaluate a summarization-based system and compare its performance against a Retrieval-Augmented Generation system for question answering. The developed summarization-based system consists of four stages: text extraction, segmentation, query-focused and hierarchical summarization, and question answering. The parameters investigated in the study includes two PDF-readers, three segment sizes, and two summary lengths. The evaluation was conducted using the metrics Exact Match and F1-score. The selected configuration of the developed system achieved an average Exact Match score of 0,81 and an average F1-score of 0,85 across three runs. The evaluated RAG configuration achieved corresponding scores of 0,76 and 0,81. The results indicate that the developed system can serve as a relevant alternative to retrieval-based strategies for question answering over financial documents.

## **Keywords**

Large Language Models, Retrieval-Augmented Generation, Question-Answering, query-focused summarization, hierarchical summarization, information-extraction, PDF-extraction, prompt engineering, Exact Match, F1-score.



# Innehållsförteckning

1	Inledning .....	1
1.1	Problemformulering .....	1
1.2	Målsättning .....	2
1.3	Avgränsningar.....	2
1.4	Författarnas bidrag till examensarbetet.....	3
1.5	Etiskt förhållningssätt .....	3
2	Teori och bakgrund.....	5
2.1	Bakgrund.....	5
2.2	Stora språkmodeller.....	6
2.2.1	Långa kontexter och centrala begrepp .....	6
2.2.2	Utmaningar med långa kontexter.....	6
2.2.3	Prompt engineering.....	7
2.3	Retrieval Augmented Generation (RAG).....	8
2.3.1	Arkitektur och funktion hos RAG .....	8
2.3.2	Chunking.....	9
2.3.3	Embeddings .....	9
2.3.4	Standardvärden för RAG-parametrar .....	10
2.3.5	Begränsningar med RAG .....	10
2.4	Alternativ till RAG.....	11
2.4.1	Prompt compression .....	11
2.4.2	Sammanfattning av kontext.....	11
2.5	Dataextraktion från dokument.....	13
2.5.1	PDF-formatet.....	14
2.5.2	Utmaningar vid textutvinning från PDF .....	14
2.6	Utvärdering av frågebesvarande och sammanfattningar .....	14
2.6.1	Utmaningar vid utvärdering av QA .....	14
2.6.2	Utvärderingsmått vid frågebesvarande.....	14
2.6.3	Utvärdering av textsammanfattningar .....	15
2.7	Externa LLM-system som referenspunkt .....	15
3	Metod.....	17
3.1	Forskningsmetodik.....	17
3.2	Systemarkitektur .....	18
3.2.1	Språkmodell.....	18
3.2.2	Textutvinning.....	18
3.2.3	Textsegmentering (chunking).....	18

3.2.4	Overlap .....	19
3.2.5	Sammanfattningar .....	19
3.3	Dataset .....	20
3.4	Experimentupplägg .....	21
3.4.1	Fas 1 – Parameterstudie .....	21
3.4.2	Fas 2 – Jämförelse mot RAG.....	22
3.5	Prompts.....	22
3.6	Genomförande.....	24
3.7	Utvärdering.....	24
4	Resultat.....	27
4.1	Utvärderingsupplägg .....	27
4.2	Resultat av parameterstudie.....	27
4.2.1	Resultat per parameter .....	29
4.2.2	Kompletterande analys av segmentstorlek .....	30
4.3	Jämförelse – QA-system vs RAG.....	31
4.3.1	Overall-resultat – QA-system vs RAG .....	31
4.3.2	Resultat per fält/fråga .....	32
4.4	Dokumentkomplexitet .....	33
5	Analys och diskussion .....	35
5.1	Analys av parameterstudien.....	35
5.2	Jämförelse mot RAG-baseline.....	37
5.3	Analys av dokumentkomplexitet .....	38
5.4	Jämförelse mot externa resultat.....	39
5.5	Begränsningar .....	39
5.5.1	Begränsningar i använda mått .....	39
5.5.2	Begränsningar i experimentupplägg.....	40
5.6	Hållbarhet, etik och samhällliga aspekter.....	41
6	Slutsatser .....	43
	Källförteckning .....	45

# 1 Inledning

I många organisationer finns större mängd information lagrat i PDF-dokument, såsom i exempelvis rapporter, kontrakt och teknisk dokumentation. Att manuellt läsa igenom och söka efter relevant information i dessa dokument kan vara både tidskrävande och ineffektivt. Med stora språkmodellers inträde så har nya möjligheter uppkommit när det kommer till att automatisera detta arbete genom att identifiera relevant information och generera svar baserat på dokumentens innehåll. För organisationer som hanterar känslig information kan användningen av externa språkmodeller däremot innebära säkerhetsrisker. Detta har ökat intresset för lokala språkmodeller, där data kan behandlas inom organisationens egna infrastrukturer. För att möjliggöra detta flöde krävs att PDF-dokument konverteras till ren text vilket medför utmaningar, såsom förlorad struktur och svårigheter i att tolka tabeller och rubriker korrekt. Vidare har språkmodeller en tendens att hallucinera samt missa information i mitten vid längre kontexter, känt som *lost in the middle*-problemet [1]. Ett viktigt steg är att få fram en kontext som är relevant, och samtidigt så pass liten att denna kan tas emot och bearbetas effektivt av en språkmodell.

## 1.1 Problemformulering

Ett vanligt tillvägagångssätt för att hantera stora mängder text och extrahera relevant information är den retrieval-baserade metoden *Retrieval-Augmented Generation (RAG)*, där relevanta textsegment identifieras och används som kontext vid frågebesvarande. Trots detta finns vissa begränsningar kopplade till hur väl relevanta textsegment kan identifieras och utnyttjas i praktiken. Forskning visar att RAG-system kan ha svårt att identifiera och extrahera relevant information, vilket leder till att svar kan bli ofullständiga, irrelevanta eller felaktiga [2]. Detta kan bli särskilt problematiskt när det kommer till hantering av ostrukturerad text som extraheras från PDF-dokument.

En annan utmaning är att långa dokument innehåller stora mängder text, samtidigt som språkmodeller har begränsningar i hur mycket information de kan hantera på samma gång. Även om vissa moderna modeller kan ta emot stora mängder kontext innebär det inte nödvändigtvis att de utifrån denna kommer kunna identifiera vilken information som är relevant för en viss fråga. Om för mycket eller irrelevant text inkluderas kan det istället försvåra för modellen att generera korrekta svar [2].

För att hantera dessa utmaningar finns ett behov av metoder som kan strukturera och reducera stora mängder text innan den bearbetas av språkmodellen. Detta inkluderar både PDF-extraktion och strategier för textkomprimering genom sammanfattning. Studien fokuserar därför på att undersöka hur olika konfigurationer av extraktion, segmentering och sammanfattning påverkar möjligheten att identifiera och extrahera relevant information ur PDF-dokument.

## 1.2 Målsättning

Målsättningen med arbetet är att utveckla en sammanfattningsbaserad pipeline och undersöka hur den påverkar *frågebesvarande (Question Answering, QA)* -prestanda över långa PDF-dokument, samt jämföra den mot ett mer etablerat tillvägagångssätt i Retrieval-Augmented Generation (RAG).

För att uppnå detta formuleras följande mål:

- Utveckla en systemprototyp för frågebesvarande över finansiella dokument. Systemet ska utföra textextrahering (ostrukturerad och strukturerad) från PDF-dokument, komprimering genom sammanfattning samt slutligt frågebesvarande utifrån den sammanfattade texten.
- Genomföra en parameterstudie för att analysera hur olika designval påverkar svars kvaliteten.
- Jämföra den föreslagna metoden med en baseline-metod, i Retrieval-Augmented Generation (RAG).
- Utvärdera resultaten med hjälp av etablerade QA-mått. I detta arbete används Exact Match (EM) och F1-värde.

Examensarbetets bidrag består av en experimentell analys av hur sammanfattningsbaserad kontext påverkar frågebesvarande, samt en jämförelse med en etablerad metod för hantering av långa dokument. Resultatet är tänkt att ge ökad förståelse för möjligheter och begränsningar med sammanfattning som alternativ till retrieval-baserade metoder. Arbetet är tänkt att även bidra med insikter kring hur dataextraktion och textstruktur påverkar efterföljande bearbetningssteg.

## 1.3 Avgränsningar

För att arbetet ska vara genomförbart inom given tidsram görs följande avgränsningar:

- Prompten hålls konstant i alla experiment. Variation av prompt-design studeras inte, då fokus ligger på hur språkmodellens output beror på kontexten den fått.
- OCR används inte vid textextrahering. De PDF-dokument som används i studien är huvudsakligen maskinläsbara. Innehåll i dokumenten som förekommer i form av logotyper, bilder eller inskannad text kommer därav inte att extraheras, vilket kan innebära att viss relevant information går förlorad som kan påverka resultatet.
- En språkmodell används genomgående i samtliga experiment för både sammanfattning och frågebesvarande. Detta görs för att isolera effekten av pipeline-design och undvika att resultatet påverkas av skillnader mellan språkmodeller.

- Datasetet begränsas till dokument inom finans-domänen, vilket innebär att resultaten inte nödvändigtvis är representativa för andra typer av dokument.
- RAG-pipelinen optimeras inte specifikt för det använda datasetet. I stället används en standardkonfiguration baserad på tidigare forskning och riktlinjer för RAG-system.

#### **1.4 Författarnas bidrag till examensarbetet**

Arbetet och samtliga ingående moment har utförts gemensamt med jämnt fördelad arbetsbelastning mellan författarna.

RAG-pipelinen som jämförs med det utvecklade systemet är skapat av författarna i tidigare kurs i mjukvarukonstruktion.

#### **1.5 Etiskt förhållningssätt**

Rapportens författare intygar härmed att rapporten är framtagen utan hjälp av generativ AI än för andra ändamål än språkkontroll och för att ta fram sökord för att hitta relaterade arbeten. Generativ AI har även använts i begränsad omfattning vid utveckling och felsökning av systemimplementationen. Exempel på prompts som använts:

- ”Varför blir parsingen fel för denna förväntade JSON-output: \*felmeddelande\*?”
- ”Visa ett exempel på ett enkelt skript för att jämföra prediktioner mot ett JSON-facit.”

Genererad kod har endast använts som stöd under utvecklingsprocessen. All kod har granskats och verifierats manuellt av rapportens författare innan implementering.



## 2 Teori och bakgrund

Detta kapitel presenterar den teori och bakgrund som ligger till grund för studien. Kapitlet behandlar centrala koncept, metoder och utmaningar relaterade till stora språkmodeller, Retrieval-Augmented Generation (RAG), kontexthantering och informationsutvinning ur PDF-dokument. Vidare presenteras utvärderingsmått för frågebesvarande, det dataset som används i studien samt tidigare resultat från externa LLM-system.

### 2.1 Bakgrund

Finansiella dokument, såsom årsredovisningar och prospekt, innehåller stora mängder information som är viktig för analys och beslutsfattande. Dokumenten är ofta långa och komplexa samt innehåller tabeller och domänspecifika begrepp. Att manuellt identifiera och extrahera specifika uppgifter ur dessa dokument kan vara både tidskrävande och resurskrävande. Samtidigt har stora språkmodeller (LLM:er) under senare år visat god förmåga inom *Natural Language Processing (NLP)* och frågebesvarande [3].

Inom finansdomänen har intresset för att använda stora språkmodeller ökat, bland annat för uppgifter såsom aktiemarknadsprognoser och frågebesvarande. Tidigare forskning visar däremot att finansiella dokument innebär särskilda utmaningar när det kommer till textutvinning eftersom information kan förekomma både i form av tabeller och i form av text [3]. Detta ställer höga krav på att rätt information kan identifieras och hämtas från dokumenten, då felaktiga svar från språkmodeller kan leda till ekonomiska konsekvenser och minskat förtroende för systemen [4].

För att förbättra språkmodellens tillgång till relevant information används ofta Retrieval-Augmented Generation (RAG), där relevanta textsegment hämtas från externa dokument och skickas vidare till språkmodellen som kontext [4]. Metoden har visat potential att förbättra precision och minska hallucinationer genom att modellen får tillgång till extern information vid generering av svar [3,4].

Samtidigt finns begränsningar med RAG-baserade system. Retrieval-steget är beroende av att semantisk sökning lyckas identifiera relevant information och relevanta textsegment, vilket påverkas av segmenteringsstrategi och *embeddings* [2]. Tidigare forskning visar även att relevant information ibland inte rankas tillräckligt högt för att inkluderas i modellens kontext, vilket kan leda till ofullständiga och felaktiga svar. Detta blir särskilt problematiskt inom domäner såsom finans, där tidigare forskning beskriver att felaktig information kan få allvarliga konsekvenser [4].

Den ökade användningen av stora språkmodeller medför även hållbarhetsrelaterade utmaningar, särskilt när det kommer till energianvändning och miljöpåverkan. Träning och användning av stora språkmodeller kommer med stora energikostnader och koldioxidutsläpp. Även om det ofta lyfts fram att det är träning som är den mest resurskrävande fasen, så pekar Wu et al. [5] på att senare studier visat att inferensfasen, det vill säga själva användandet av modeller i applikationer och mer specifikt storskalig användning, snarare är det som ger störst energiavtryck. Studien visar till

exempel på att användandet av *ChatGPT* under ett år har 25 gånger större växthusgasutsläpp än vad det kostade att träna GPT-3:s modell [5].

Jegham et al. [6] beskriver att energiförbrukningen vid inferensfasen av stora språkmodeller är kopplad till både mängden input som behöver bearbetas och mängden output som genereras vid svar. Studien visar hur längre prompts ökar bearbetningskostnaden och latens, samtidigt som större mängder genererad text också är förknippade med ökad energiförbrukning.

Mot bakgrund av de utmaningar som finansiella dokument medför samt de begränsningar som finns i retrieval-steget i RAG-baserade system undersöker denna studie en flerstegspipeline för dokumentbaserat frågebesvarande över finansiella dokument. Metoden jämförs därefter med ett traditionellt RAG-system.

## 2.2 Stora språkmodeller

Stora språkmodeller (LLM:er) används i allt större utsträckning för uppgifter som involverar bearbetning av omfattande textmängder och generering av svar. Inom dokumentbaserat frågebesvarande har modellens förmåga att hantera lång kontext stor betydelse för resultatet.

### 2.2.1 Långa kontexter och centrala begrepp

En central egenskap hos stora språkmodeller är deras förmåga att hantera en viss mängd text, ofta definierad i antal *tokens*. Tokens används som ett mått för hur text representeras och bearbetas av språkmodeller. En token kan motsvara ett helt ord, delar av ett ord eller enstaka tecken beroende på språk. För engelska är det ungefärliga måttet att 1 token = 4 tecken [7]. Kontexten representerar den text som modellen har tillgång till vid generering av ett svar och består vanligtvis av både indata, såsom fråga och bakgrundstext, samt tidigare genererad text. Storleken på denna kontext begränsas av modellens så kallade *context window*, vilket anger det maximala antal tokens som kan bearbetas vid ett och samma tillfälle [8]. Utvecklingen av moderna språkmodeller har lett till en successiv ökning av detta kontextfönster, vilket möjliggör att större delar av dokument kan inkluderas direkt i prompten. Genom att inkludera hela dokument i prompten elimineras behovet av att i förväg selektera information. Vid andra metoder såsom vid RAG där svarsgenerering föregås av ett retrieval-steg, kan relevant information missas att ta med, vilket kan försvåra språkmodellens förmåga att senare generera korrekta svar [2].

### 2.2.2 Utmaningar med långa kontexter

Trots språkmodellens successivt ökande kontextstorlek har tidigare forskning visat att språkmodeller har begränsningar i hur effektivt de kan utnyttja denna. Liu et al. [1] genomför en systematisk studie av hur språkmodeller använder information vid långa indata. Studien analyserar modellernas prestanda i uppgifter som kräver att relevant information identifieras inom en större kontext. Genom att kontrollera både kontextens längd och positionen för den relevanta informationen visar författarna att modellernas prestanda påverkas kraftigt av var i kontexten informationen är placerad.

Resultaten visar att språkmodeller uppvisar en tydlig U-formad prestandakurva beroende på positionen av relevant information i kontexten [1]. Modellerna presterar

generellt bäst när den relevanta informationen återfinns i början eller slutet av kontexten, medan prestandan försämras när informationen är placerad i mitten. Detta fenomen benämns *“lost in the middle”* och beskriver modellernas begränsade förmåga att identifiera och använda information som befinner sig i mitten av långa texter. Det noterades även att modeller i vissa fall presterar sämre när relevant information placeras i mitten av kontexten än när ingen extern kontext tillhandahålls alls [1].

Vidare observerar Liu et al. [1] att användning av längre kontext innebär en avvägning mellan tillgång till mer information och modellens förmåga att effektivt utnyttja denna information. Prestandan kan initialt förbättras när mer kontext tillförs, men förbättringen avtar snabbt och en platå nås trots att ytterligare information inkluderas. Resultaten indikerar att språkmodeller har en begränsad förmåga att tillgodogöra sig ytterligare kontext, och att en ökning av mängden indata inte nödvändigtvis leder till motsvarande förbättringar i svars kvaliteten.

### 2.2.3 Prompt engineering

Natural Language Processing (NLP) är ett område inom artificiell intelligens och lingvistik som handlar om att få datorer att förstå, analysera och skapa mänskligt språk. Tekniken används i många tillämpningar, såsom informationsutvinning, sammanfattning och frågebesvarande [9]. *Prompt engineering* är en metod inom NLP som används för att styra och förbättra stora språkmodellers förmåga att förstå och generera text genom att optimera de instruktioner som ges till en modell. Genom att formulera tydliga och strukturerade prompts kan modellen tolka uppgiften på ett bättre sätt och generera mer relevanta och korrekta resultat. Yao et al. [10] beskriver hur strukturerad promptdesign kan förbättra informationsutvinning ur dokument. Studien visar att språkmodellers precision och robusthet kan förbättras genom att prompts utformas med tydliga instruktioner, specificerade outputformat samt begränsningar.

Vid utformning av prompts kan olika prompting-strategier användas för att påverka hur modellen tolkar och genomför en uppgift. Sahoo et al. [11] beskriver prompt engineering som en metod för att styra stora språkmodeller genom instruktioner och kontext utan att modellens interna parametrar behöver förändras. Vidare visas att språkmodellers prestation påverkas av hur prompts struktureras och att välutformade prompts kan förbättra modellens precision och informationsutvinning vid komplexa uppgifter. Samtidigt beskriver författarna bland annat *zero-shot* och *few-shot* prompting som två vanliga strategier inom prompt engineering. *Zero-shot* prompting innebär att modellen utför en uppgift utan tidigare exempel, medan *few-shot* prompting innebär att modellen vägleds genom ett antal exempel på önskad input och output för att förbättra modellens förståelse för uppgiften [11].

OpenAI och Microsoft har även lyft fram praktiska riktlinjer för hur prompts kan utformas för att förbättra språkmodellers precision. Dessa inkluderar bland annat tydliga instruktioner, specificerade outputformat, samt uppdelning av komplexa uppgifter i mindre delmoment [8,12,13]. OpenAI beskriver även hur olika roller och instruktionstyper kan användas för att styra modellens beteende och responsgenerering. Genom att definiera en tydlig identitet eller övergripande uppgift för mo-

dellen kan instruktionerna göras mer uppgiftsspecifika vilket kan bidra till mer relevanta och konsekventa svar från modellen [8]. Microsoft beskriver även hur språkmodeller kan styras genom rollbaserade instruktioner, där modellen tilldelas ett specifikt syfte eller ansvarsområde, exempelvis ifall det är i form av en analys eller informationsbearbetning, för att påverka hur svar genereras och struktureras [13].

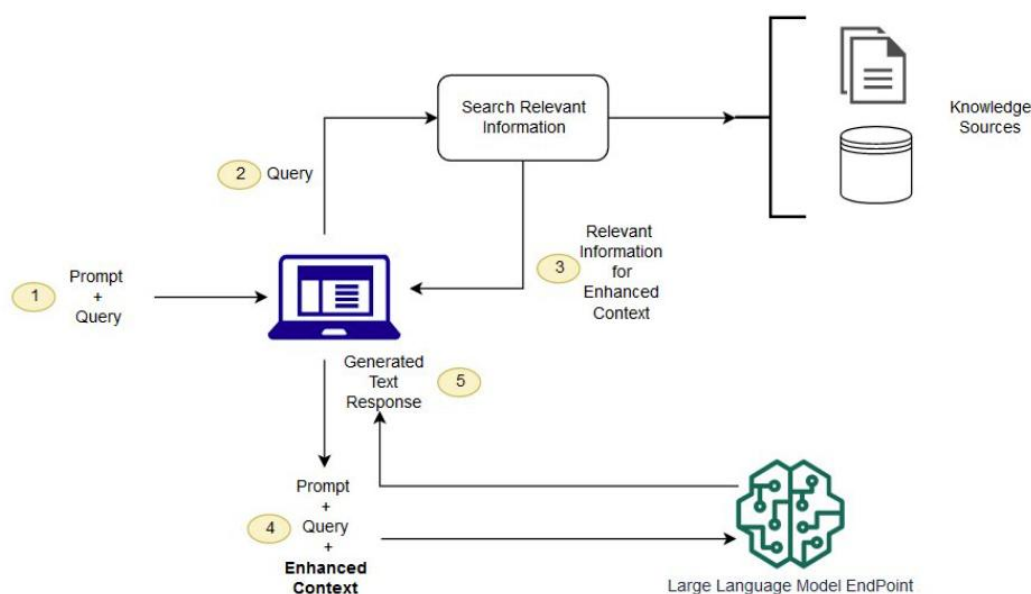
Vid informationsutvinning används prompt engineering för att få modellen att identifiera och extrahera specifika delar av information av text. Microsoft framställer även hur komplexa uppgifter kan delas upp i mindre steg genom stegvis prompting för att förbättra språkmodellens precision i genererade svar [13].

## 2.3 Retrieval Augmented Generation (RAG)

För att hantera begränsningar hos språkmodeller vid användning av långa kontexter har retrieval-baserade metoder utvecklats. En framträdande sådan metod är Retrieval-Augmented Generation (RAG), där en språkmodell kompletteras med hämtning av extern relevant information.

### 2.3.1 Arkitektur och funktion hos RAG

RAG kombinerar en språkmodell med ett externt kunskapslager genom att först hämta relevanta dokument och därefter använda dessa som kontext vid generering av svar [14]. RAG använder både parametrisk och icke-parametriskt minne. Den parametriska delen utgörs av en förtränad språkmodell, medan den icke parametriska delen består av ett dokumentindex, vanligtvis uppbyggt av en stor samling textdokument, som delas upp i mindre textsegment och representeras som vektorer för att möjliggöra effektiv sökning [14]. RAG-modellen består av två huvudsakliga komponenter: en *retriever* och en *generator*, vilket illustreras i figur 2.1. Frågan skickas först till retrievern som används för att, givet frågan, identifiera de mest relevanta dokumenten från det externa kunskapslagret. Därefter använder generatoren dessa dokument tillsammans med frågan för att generera ett svar. Detta möjliggör att modellen kan basera sina svar på aktuell och explicit tillgänglig information, till skillnad från enbart på vad som lagrats i modellens parametrar.



Figur 2.1: Visar flödet i en RAG-pipeline [15].

Lewis et al. [14] visar genom experiment att RAG-modeller uppnår stark prestanda på kunskapsintensiva uppgifter, särskilt inom open-domain frågebesvarande, där de överträffar både parametriska språkmodeller och tidigare retrieval-baserade metoder. Vidare observeras att modellerna genererar mer faktabaserade och specifika svar, vilket minskar förekomsten av hallucinationer jämfört med modeller som enbart förlitar sig på intern kunskap [14].

### 2.3.2 Chunking

För att möjliggöra effektiv hantering av stora dokument i retrieval-baserade system delas texten ofta upp i mindre segment, så kallade chunks, som behandlas som separata enheter vid indexering och informationssökning. Varje segment representeras individuellt och används vid retrieval-steget för att identifiera relevant kontext. En vanlig segmenteringsstrategi är så kallad *fixed-size chunking*, där text delas upp i segment av fast storlek baserat på antal token eller tecken, vilket ger en enkel och effektiv implementering. Däremot, då segmenteringen inte sker med hänsyn till semantiska strukturen i texten kan relaterad information delas upp mellan flera segment, vilket kan leda till fragmenterad kontext och försämrad kvalitet vid retrieval [16]. För att hantera denna begränsning har mer avancerade segmenteringsstrategier utvecklats. En sådan metod är *semantic chunking*, där text delas upp baserat på semantisk betydelse snarare än fasta längder, vilket kan bidra till att bevara samband inom segmenten. En annan strategi är *proposition-based chunking*, där text bryts ner i mindre atomära faktaenheter, vilket kan förbättra precisionen vid retrieval [17]. Utöver segmenteringsstrategi spelar även *overlap* roll. Overlap är ett förbestämt antal tokens som överlappar mellan två efterföljande segment, den sista delen i det första segmentet förekommer igen i början av nästa segment [18]. Syftet med overlap är att minska risken för att viktig kontext går förlorad mellan segment, vilket kan ske då relaterad information delas upp mellan olika textsegment. En systematisk studie av segmentering i RAG visar att overlap mellan segment däremot inte ger någon mätbar förbättring i svars kvalitet, samtidigt som det ökar lagring- och beräkningskostnader [19].

### 2.3.3 Embeddings

*Embeddings* utgör en central komponent vid RAG och används för att representera text som numeriska vektorer i ett semantiskt rum. I stället för att behandla ord som diskreta enheter möjliggör denna representation att man kan fånga semantiska relationer mellan ord geometriskt. Mikolov et al. [20] visar att ord med liknande betydelse placeras nära varandra i vektorrummet, och att semantisk likhet mellan ord kan mätas genom beräkning av avstånd mellan varandra i detta vektorrum. Vidare har senare forskning utökat denna princip till att omfatta hela meningar och längre textsegment. Reimers et al. [21] presenterar Sentence-BERT, en metod som genererar semantiskt meningsfulla embeddings för meningar, vilket möjliggör att likheten och relationer mellan textsegment kan beräknas effektivt med exempelvis cosinuslikhet. Inom RAG-system används embeddings för att representera både användarfrågor och dokument i samma vektorrum. Genom att jämföra dessa representationer kan relevanta textsegment identifieras, det vill säga segment vars vektorer ligger nära frågans vektor i vektorrummet, och därefter användas som kontext när språkmodellen genererar ett svar. Vidare har flerspråkiga embeddingmodeller utvecklats för att möjliggöra semantisk retrieval över flera språk genom att representera text från olika språk i ett gemensamt semantiskt vektorrum. Wang et al. [22] beskriver

hur embeddings utgör en central komponent inom retrieval, samtidigt som många tidigare embeddingmodeller huvudsakligen tränats på engelskspråkig data. För att möjliggöra semantisk retrieval i flerspråkiga miljöer presenterar författarna från Microsoft de flerspråkiga E5-modellerna, vilka bygger på XLM-RoBERTa. Modellerna har tränats på stora mängder flerspråkiga textpar för användning inom semantisk sökning över flera språk [22].

#### 2.3.4 Standardvärden för RAG-parametrar

Tidigare forskning visar att segmentstorlek vid fixed-size chunking påverkar retrievalprestanda beroende på dokumenttyp och frågekaraktär. Mindre segmentstorlekar, mellan 64-128 tokens, har visat god prestanda för dataset med korta och fakta-baserade svar där relevant information är lokalt koncentrerade i texten. Samtidigt har större segmentstorlekar, mellan 512-1024 tokens, visat förbättrad prestanda för dokument och frågor som kräver bredare kontextuell förståelse [23]. En segmentstorlek på 512 tokens förekommer även som standardvärde i NVIDIAs RAG-dokumentation, där större segmentstorlekar beskrivs som fördelaktigt för att bevara mer kontextuell information, även om detta samtidigt kan öka mängden irrelevant kontext [24]. När det gäller överlap rekommenderar Microsoft generellt ett värde omkring 10-25% av segmentstorleken för att bibehålla kontextuell kontinuitet utan att skapa alltför stor redundans [25]. NVIDIA använder ett standardvärde på 150 tokens överlap tillsammans med segmentstorleken 512 tokens i sin RAG-dokumentation [24].

Två andra parametrar som ofta används vid utformning av RAG-system är retrievaldjup (*top-k*) och *similarity threshold*. Retrievaldjup, ofta benämnt som top-k, anger hur många textsegment som hämtas från vektordatabasen och skickas vidare till språkmodellen för att användas vid kontext vid generering av svar. Bhat et al. [23] använde top-k-värden mellan 1-5 vid en undersökning av segmentstorlekars påverkan på retrievalprestanda. Similarity threshold används för att filtrera bort retrievalresultat med låg semantisk relevans. Radeva et al. [26] visar att threshold-värden mellan omkring 0,55-0,65 fungerade som bäst för de utvärderade modellerna i deras RAG-miljö.

#### 2.3.5 Begränsningar med RAG

Retrieval-baserade metoder medför samtidigt flera utmaningar. RAG-system är beroende av att relevanta dokument faktiskt kan identifieras och hämtas under retrieval-steget. Om relevanta dokument inte återfinns bland de högst rankade resultaten, eller om irrelevanta dokument inkluderas, kan detta leda till felaktiga eller ofullständiga svar från språkmodellen som ska besvara frågor baserat på kontexten [2]. Utöver retrieval-steget visar forskning att RAG-system kan generera svar som är felaktiga trots att svaren ska baseras på de relevanta källor som hämtats under retrieval. En empirisk studie av Magesh et al. [27] visar att RAG-baserade system fortfarande uppvisar hallucinationer i 17-33 % av fallen. Detta indikerar att tillgång till relevanta dokument inte garanterar korrekta och tillförlitliga svar. Vidare kan problem i RAG-system även uppstå i samspelet mellan retrieval och generering. Språkmodellen kan producera svar som refererar till existerande källor utan att dessa faktiskt stödjer de påståenden som görs, vilket kan leda till missvisande och felaktigt underbyggda svar

[27]. Denna typ av fel blir problematisk då svar kan framstå trovärdiga genom inkludering av referenser.

## 2.4 Alternativ till RAG

Som alternativ till RAG har flera andra metoder utvecklats som i stället fokuserar på att förbättra hur kontext bearbetas innan den används i prompten till modellen, eller på att förbättra själva prompten och dess instruktioner. Den senare metoden tillhör så kallade prompt-centrerade metoder, vilka syftar till att förbättra effektiviteten hos stora språkmodeller genom att minska komplexiteten i indata snarare än att förändra själva modellen [28].

### 2.4.1 Prompt compression

En central metod inom detta område är *prompt compression*, där målet är att begränsa längden på prompten samtidigt som relevant information bevaras. Detta kan vara viktigt eftersom långa prompts kräver mer minne, ökar beräkningskostnaden och kan försämra svarstider och användarupplevelse [28]. Prompt compression kan delas in i två olika angreppssätt, *hard prompts* och *soft prompts*. Hard prompt-metoder innebär att mindre relevant eller redundant text tas bort för att göra prompten mer kompakt [28]. Soft prompt-metoder innebär i stället att prompten representeras i komprimerad form genom vektorrepresentationer [28]. Fördelarna med prompt compression är att beräkningskostnader kan reduceras och svarshastigheten förbättras, eftersom modellen behöver bearbeta färre tokens. Samtidigt finns vissa nackdelar och begränsningar. En central utmaning är risken för informationsförlust, där viktig information kan försvinna eller förvrängas vid komprimering av text [28]. Metoden kan även öka systemets komplexitet, eftersom ytterligare bearbetningssteg krävs jämfört med att skicka prompten direkt till modellen. Detta kan innebära att den totala beräkningskostnaden och bearbetningstiden i vissa fall blir likvärdig med användning utan kompression [28]. Prompt compression liknar i viss mån sammanfattning av kontext, men skiljer sig genom att fokus ligger på att komprimera själva instruktionerna till modellen snarare än den kontext som modellen ska basera sitt svar på. Metoderna kan användas i kombination för att reducera det totala antalet tokens som skickas till modellen.

Det kan intuitivt antas att prompt compression skulle leda till mer hållbar användning av språkmodeller då det skulle innebära mindre kontext. Senare forskning visar däremot att detta samband är mer komplext. Johnson [29] visar att prompt compression i vissa fall kan leda till en kraftig ökning av modellens genererade output, vilket ökar den totala beräkningskostnaden och därmed även energiförbrukningen.

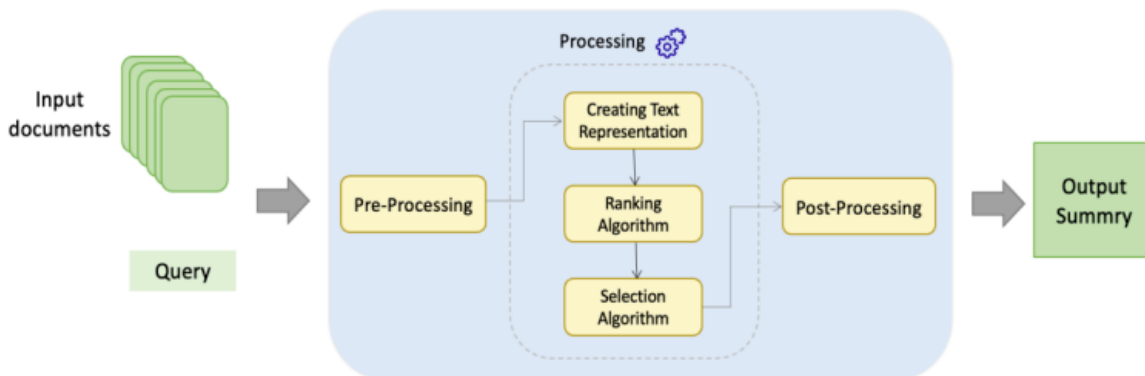
### 2.4.2 Sammanfattning av kontext

Ett annat alternativ till RAG är att sammanfatta kontexten innan den skickas till modellen, i stället för att använda den ursprungliga textmängden. Detta kan genomföras på flera olika sätt, där frågebesvarande därefter utförs på den sammanfattade kontexten för att generera svar.

#### 2.4.2.1 Frågefokuserad sammanfattning

Frågefokuserad sammanfattning (*query-focused summarization*) skiljer sig från generell sammanfattning genom att sammanfattningen genereras utifrån en specifik

fråga från användaren snarare än hela dokumentets innehåll. Detta medför att endast information som är relevant för frågan inkluderas i sammanfattningen [30]. Processen för query-focused summarization består oftast av flera steg, vilket illustreras i figur 2.2. Först sker *pre-processing*, där både dokumentet och frågan behandlas med målet att reducera brus och minska beräkningstiden. Därefter representeras texten i en form som möjliggör analys, och meningarna i texten rankas sedan utifrån deras relevans för frågan. Till sist sker *post-processing* där överflödigt information tas bort eller meningar förkortas [30].



Figur 2.2: Visar hur den generella arkitekturen för frågefokuserade sammanfattningar ser ut [30].

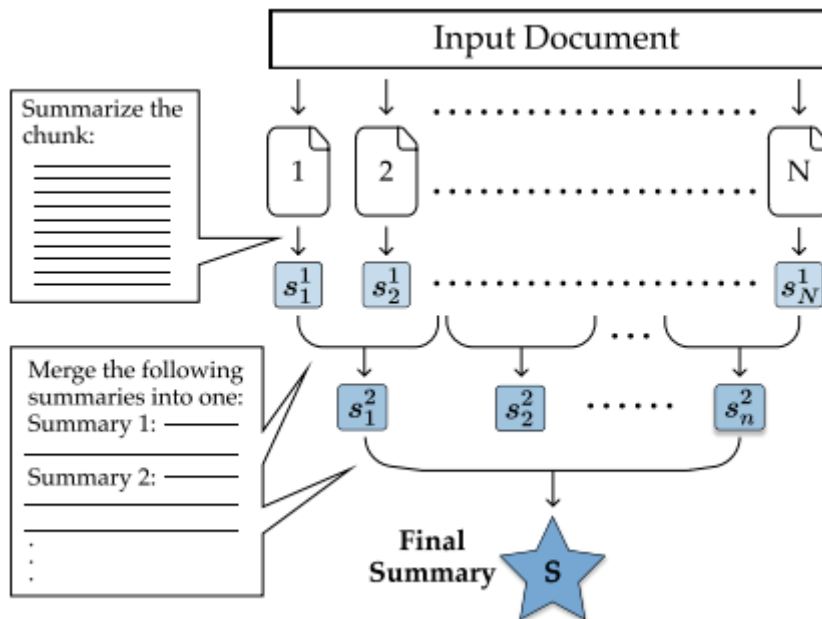
Alanzi et al. [30] framhåller att metoden är särskilt effektiv för att generera sammanfattningar som adresserar användarens frågor och tillgodoser för deras informationsbehov. Samtidigt innebär detta fokus att metoden kan vara mindre lämplig för att skapa en heltäckande sammanfattning av ett dokument, då information som inte är direkt relaterad till frågan riskerar att exkluderas.

#### 2.4.2.2 Användning av frågebesvarande (QA) för att förbättra sammanfattning

Som ett alternativ introducerar Sinha [31] en metod för att summera genom att använda QA-prompting, en metod där QA ligger som ett mellansteg innan själva sammanfattningen genereras. QA-prompting bygger på idén att en språkmodell först får besvara ett antal relevanta frågor om texten, och därefter använda både originaltexten och de genererade svaren som kontext för att skapa en sammanfattning. Sinha [31] visar att det genom QA-prompting går att uppnå 29 % bättre sammanfattningar i ROUGE-värde än vanliga sammanfattningar utan QA, vilket visar att metoden är bra på att fånga upp viktig information. Vidare har metoden visat god förmåga att hantera *positional bias* genom att relevant information först extraheras via frågor, vilket innebär att relevant innehåll placeras närmare modellens senaste kontext. Detta minskar risken för att information i mitten av texten ignoreras [31]. Metoden är dessutom relativt enkel och kräver endast ett anrop till modellen, vilket gör att den är beräkningsmässigt effektiv jämfört med mer komplexa metoder. Sinha [31] diskuterar samtidigt vissa begränsningar med metoden. Bland annat krävs relevanta frågor som ofta behöver definieras manuellt, vilket kan innebära extra arbete och behov av domänkunskap. Metoden är även mindre effektiv för språkmodeller med begränsad kontextstorlek, eftersom kontextlängden fortfarande kan bli omfattande.

### 2.4.2.3 Hierarkisk sammanfattning

Hierarkisk sammanfattning är en metod för att hantera långa dokument genom att dela upp texten i mindre textsegment som sammanfattas stegvis. Wu et al. [32] lyfter fram hur långa dokument delas upp i mindre delar som individuellt sammanfattas, och att dessa delsammanfattningar sedan kombineras och sammanfattas igen stegvis i nivåer tills en slutgiltig sammanfattning av dokumentet genererats, som framgår av figur 2.3.



Figur 2.3: Illustrerar hur textdokument delas in i segment och sammanfattas hierarkiskt tills det att den når en *final summary* [33].

Metoden är fördelaktig då den möjliggör hantering av dokument som annars överskrider språkmodellens kontextfönster. Då varje steg och nivå behandlar en begränsad mängd text kan modellen successivt bygga upp en mer kompakt representation av hela dokumentet. En nackdel med denna metod är däremot att varje steg baseras på lokal kontext från steget innan. Om information förloras tidigt i processen riskerar dessa fel att ackumuleras genom de efterföljande sammanfattningsstegen. Senare forskning har undersökt en vidareutveckling av metoden genom att titta på hur delsammanfattningar kan kombineras på ett mer effektivt sätt. Förbättringar föreslås i sammanslagningssteget, där ytterligare kontext kan integreras för att minska risken för informationsförlust och förbättra korrektheten i den slutgiltiga sammanfattningen [33]. Ou et al. [33] delar upp dokument i segment om 8000 tokens som därefter sammanfattas hierarkiskt.

## 2.5 Dataextraktion från dokument

Dataextraktion från dokument är ett centralt steg i många textbaserade system, där dokument behöver omvandlas till text för att kunna användas i vidare bearbetning.

### 2.5.1 PDF-formatet

PDF är ett filformat, från början skapat av Adobe, som gör det enkelt att visa och dela dokument och som är plattformsoberoende. PDF står för “*Portable Document Format*” och är idag ISO-standardiserat. Formatet kan innehålla text, bilder, länkar och skannad text, vilket kan försvåra extraktion av all relevant information. [34]

### 2.5.2 Utmaningar vid textutvinning från PDF

Att extrahera text från PDF-dokument kan vara utmanande då formatets struktur ofta är komplex. Myasoedov et al. [35] redogör för hur PDF saknar en tydlig logisk struktur och där innehåll i stället bevaras utifrån visuell layout. Detta gör det svårt att tolka och återskapa element såsom stycken, rubriker, listor, tabeller och kolumner [35]. Vidare så diskuteras även att en ytterligare utmaning är att PDF-dokument kan förekomma i olika former, där en del innehåller maskinläsbar text medan andra kan vara inskannade och därför bestå av endast bilder. Vid inskannade PDF-dokument krävs optisk teckenigenkänning (OCR) vilket ställer andra krav på processen, särskilt vid låg bildkvalitet eller om det förekommer brus.

Studier har visat att kvaliteten på PDF-extraktion kan påverka prestandan i efterföljande RAG-och QA-system. Lin [36] beskriver likt Myasoedov et al. [35] hur PDF-filer ofta saknar tydlig strukturell information, vilket kan leda till informationsförlust vid textutvinning och segmentering. Studien jämför mer ostrukturerad textutvinning med *PyPDF* mot mer strukturbevarande parsingmetoder och visar att kvaliteten på PDF-parsing påverkar hur väl relevant information senare kan hämtas och användas vid frågebesvarande. Inom LangChain finns stöd för flera olika PDF-läsare med varierad grad av strukturbevarande extraktion, däribland *PyPDF* som lyfts i studien av Lin [36] samt *PyMUPDF4LLM* som möjliggör extraktion i *markdown*-format [37]. Markdown-formatet möjliggör att struktur i viss utsträckning kan bevaras för vidare bearbetning. Både *PyPDF* och *PyMuPDF4LLM* är open source-verktyg.

## 2.6 Utvärdering av frågebesvarande och sammanfattningar

Frågebesvarande är en central uppgift inom NLP där en modell, givet en fråga och en tillhörande kontext, genererar ett svar som är relevant och korrekt i förhållande till den givna texten. I moderna språkmodeller formuleras denna typ av uppgift ofta som ett text-till-text problem, där indata (fråga och kontext) och utdata (svar) representeras i form av textsekvenser [38].

### 2.6.1 Utmaningar vid utvärdering av QA

Utvärdering av QA-system är förknippad med flera utmaningar. En central svårighet är att ett korrekt svar ofta kan uttryckas på flera sätt, exempelvis genom variation i ordval och grammatik. Detta innebär att en strikt jämförelse mellan en prediktion och ett svar inte alltid ger en rättvis bild av modellens faktiska prestation. Vidare kan svar bestå av ett eller flera textsegment, vilket kan komplicera utvärdering ytterligare. I dessa fall kan ett svar vara delvis korrekt även om det inte exakt matchar referenssvaret.

### 2.6.2 Utvärderingsmått vid frågebesvarande

För att utvärdera QA-system används i stor utsträckning standardiserade mått, där två av de mest etablerade är *Exact Match (EM)* och *F1-värde*. Dessa mått används bland annat i datasetet *SQuAD* introducerat av Rajpurkar et al. [39]. *Exact Match*

mäter andelen prediktioner som exakt matchar referenssvaret. Måttet är strikt och ger endast utslag om det predikterade svaret är identiskt med referensen, utan hänsyn till semantiska variationer eller alternativa formuleringar. F1-värde är ett överlappsbaserat mått som tar hänsyn till både precision och recall. Inom QA används F1-värde för att mäta graden av överlapp mellan det predikterade och det korrekta svaret. Detta kan köras på token-nivå. Detta innebär att måttet kan ge delvis poäng till svar som är delvis korrekta.

Utöver automatiska utvärderingsmått förekommer även kvalitativa mått inom NLP och QA, såsom mänsklig bedömning. Tidigare forskning [40] har argumenterat för att automatiska mått främst mäter statistisk överensstämmelse, medan mänsklig utvärdering bättre kan fånga aspekter såsom relevans, begriplighet och praktisk korrekthet. Till skillnad från automatiska mått möjliggör mänsklig utvärdering en mer nyanserad bedömning av genererade svar och kan därmed identifiera svar som är korrekta och användbara trots att de inte exakt överensstämmer med ett referenssvar.

### 2.6.3 Utvärdering av textsammanfattningar

För utvärdering av automatiskt genererade sammanfattningar används ofta överlappsbaserade mått, där ett av de mest etablerade är *ROUGE* (Recall-Oriented Understudy for Gisting Evaluation). *ROUGE* är ett lexikalt mått som jämför en genererad sammanfattning med en eller flera referenssammanfattningar genom att mäta graden av överlappning mellan texterna [41]. *ROUGE* kan beräknas i olika varianter beroende på hur överlappet definieras, exempelvis baserat på enskilda ord (*ROUGE-1*), sekvenser av ord (*ROUGE-2*) eller gemensamma delsekvenser mellan texter (*ROUGE-L*). Måttet används i stor utsträckning inom forskning kring textsammanfattning och utgör en standardmetod för att evaluera sammanfattningar. Samtidigt har användningen av utvärderingsmetoder förskjutits från traditionella överlappsbaserade mått, såsom *ROUGE*, mot mer semantiskt orienterade mått, såsom *BERTScore* och *MoverScore*. Dessa har dock fortfarande utmaningar, bland annat relaterat till tolkningsbarhet och beräkningskostnader [41].

## 2.7 Externa LLM-system som referenspunkt

Dokumentuppsättningen som används i denna studie har tidigare testats manuellt av uppdragsgivaren med externa LLM-systemen ChatGPT, Le Chat och Gemini. Resultaten används som kompletterande referenspunkt vid senare analys av det utvecklade systemets prestanda i relation till externa kommersiella språkmodeller. Vid testningen användes motsvarande frågor och samma förklarande information kring respektive fält som i denna studie. Resultaten från de externa systemen baseras på manuell bedömning, där mindre variationer i exempelvis stavning och formatering accepterades som korrekta svar. Resultaten presenteras i tabell 2.1.

Tabell 2.1: Tidigare manuella resultat Exact Match från externa LLM-system på den använda dokumentuppsättningen.

System	EM
ChatGPT	0,88
Le Chat	0,78
Gemini	0,59



## 3 Metod

I detta arbete har en litteraturstudie genomförts i syfte att skapa en förståelse för relevanta metoder och etablerade tillvägagångssätt inom området. Med utgångspunkt i litteraturstudien identifierades särskilt PDF-extraktion, segmentering och sammanfattningsstrategi som återkommande relevanta designval i tidigare forskning. Experimenten utformades därför kring tre parametrar: val av PDF-läsare, segmentstorlek samt sammanfattningslängd, medan segmenteringsstrategi och sammanfattningsstrategi hölls konsekvent genom samtliga experiment. Parametrarna undersöks genom en metod som består av fyra huvudsakliga steg: dataextrahering, textsegmentering, sammanfattning och frågebesvarande.

### 3.1 Forskningsmetodik

Det första steget utgörs av dataextrahering, där PDF-dokument omvandlas till extraherbar text. Eftersom datasetet i denna studie består av dokument med huvudsakligen maskinläsbar text tillämpas inte OCR i experimenten. Två olika PDF-läsare används, båda avsedda för extraktion av maskinläsbar text. De två PDF-läsare som valts är PyPDF, som genererar ostrukturerad text, och PyMuPDF4LLM, som genererar mer strukturerad text genom att bevara bitar av textens ursprungliga struktur, såsom rubriker och tabeller. Genom att jämföra dessa två PDF-läsare möjliggörs en analys av hur viktigt det strukturella bevarandet är för efterföljande steg i det utvecklade QA-systemet.

Det andra steget utgörs av textsegmentering där texten delas upp i bitar av fast storlek. Detta steg behövs då forskning [1] visat att stora språkmodeller kan ha problem vid större kontexter och missa information som ligger i mitten samt att språkmodeller har ett begränsat kontextfönster. Samtidigt kan för lite kontext innebära att sammanhang som hade behövts för att besvara användarens fråga går förlorat. Studien undersöker tre olika segmentstorlekar, samtliga utefter fixed-size chunkingstrategi. De storlekar som används är en mindre (1000 tokens), en större (8000 tokens), samt en mellanliggande (4000 tokens). Syftet är att undersöka segmentstorlekens påverkan på QA-prestandan.

Det tredje steget utgörs av skapandet av sammanfattningar. Sammanfattningsstrategin som används är hierarkisk sammanfattning i kombination med frågefokuserad sammanfattning. Den parameter som valts att varieras i experimenten är sammanfattningarnas maxlängd. Storlekarna som valts att testas är 750 och 1500 tokens.

Det fjärde steget är frågebesvarande (QA). Frågorna skickas tillsammans med den slutgiltiga sammanfattningen till den valda språkmodellen för att besvara frågorna baserat på sammanfattningen. En mer detaljerad genomgång av respektive parameter med motiveringar presenteras i delkapitlet systemarkitektur.

Efter att en parameterstudie genomförts där samtliga 12 konfigurationer testats två gånger vardera, jämförs den bäst presterade konfigurationen mot ett RAG-system. För respektive metod genomförs tre separata körningar för att kunna analysera båda metodernas prestanda.

## 3.2 Systemarkitektur

Systemet består av en pipeline i flera steg, inklusive textutvinning, segmentering, sammanfattning och frågebesvarande. Utformningen av denna pipeline innebär ett antal designval och parametrar. Efter sista steget genomförs även ett utvärderingssteg, där resultaten jämförs mot ett fördefinierat facit. I följande avsnitt beskrivs de val som gjorts i denna studie.

### 3.2.1 Språkmodell

En språkmodell används för både sammanfattning och frågebesvarande. Modellen hålls konstant genom samtliga experiment för att isolera effekten av övriga parametrar. Samma språkmodell används senare även vid jämförelsen mot RAG-pipeline i fas 2 (Se avsnitt 3.4.2). Detta möjliggör analys av hur förändringar i det utvecklade QA-systemet påverkar resultaten utan att resultaten påverkas av skillnader mellan olika modeller. I arbetet användes *mistral-small3.2:24b*. Modellen valdes eftersom den bedömdes erbjuda en lämplig balans mellan stöd för de kontextstorlekar som studien undersökte, flerspråkig textbearbetning och generell prestanda. Vid anrop till språkmodellen sattes *temperature* till 0.0 för att minska variation i genererade svar och för att göra modellens output mer deterministisk. Detta ligger i linje med Microsofts rekommendation om att använda låg temperature för uppgifter där fokus, stabilitet och konsekventa svar prioriteras [13].

### 3.2.2 Textutvinning

Text extraheras från PDF-dokument med hjälp av två olika metoder med olika grad struktur. Syftet med detta är att undersöka hur kvaliteten på textutvinning påverkar efterföljande steg i pipelinen. Den ostrukturerade metoden extraherar text utan att bevara dokumentets layout och semantiska struktur. Det valda verktyget för ostrukturerad textextrahering är PyPDF, vilket även används i studien av Lin [36] vid jämförelse mot mer strukturbevarande parsingmetoder. Den valda metoden för strukturerad output är PyMuPDF4LLM som ger strukturerad markdown och bevarar rubriker, vilket potentiellt kan förbättra modellens förståelse av innehållet. Båda dessa PDF-läsare finns listade i LangChains dokumentation över alternativ för att läsa och bearbeta PDF-filer, där PyMuPDF4LLM är den enda PDF-läsaren som dokumentationen uttryckligen nämner genererar markdown-output [37]. Genom att jämföra dessa två metoder möjliggörs en analys av hur olika nivåer av strukturell information i indata påverkar systemets QA-prestanda.

### 3.2.3 Textsegmentering (chunking)

För att dela upp dokumentet i textsegment valdes fixed-size chunking som segmenteringsstrategi, då denna metod återkom som en central del i flera av de studier som analyserades i litteraturstudien. Detta gav tydligt underlag för att undersöka hur olika segmentstorlekar påverkar det utvecklade QA-systemets prestanda. Genom att använda fixed-size chunking möjliggörs även en mer kontrollerad analys, eftersom segmenteringsstrategin kan hållas konstant mellan experimenten, både för det utvecklade QA-systemet och för RAG-pipelinen. Att fastställa en optimal segmentstorlek är utmanande då tidigare forskning visar att det inte finns en universal bästa storlek. Det gäller att göra en avvägning mellan att bevara semantisk sammanhängande information och begränsa totala mängden indata. För stor kontext kan enligt Liu et al. [1] leda till försämrad prestanda vilket visas på i studien om *lost in the*

*middle*-problemet. Liu et al. [1] har valt att använda en segmentstorlek om 100 tokens per segment. Segmenten benämner de som dokument. De varierar storleken på totala kontexten genom att lägga ihop flera segment som blir input-kontexten till modellen. De testar 10, 20 och 30 i hopslagna dokument vilket ger 1000, 2000 och 3000 tokens. Ou et al. [33] använder i stället segment på 8000 tokens för att göra en hierarkisk sammanfattning där de delar in stora dokument i segment om 8000 tokens för att sedan sammanfatta dessa med hjälp av en språkmodell. Sedan slås flera segment ihop tills de når önskad segmentstorlek för att sedan sammanfattas igen, i nivåer, tills det att en sista *final summary* nåtts.

Mot denna bakgrund, och för att kunna analysera hur segmenteringsnivå påverkar prestandan har vi valt tre olika segmentstorlekar: 1000, 4000, 8000 tokens. Storlekarna 1000 och 8000 tokens representerar två tydligt skilda nivåer av segmentering som förekommer i den forskning som beskrivits ovan. För att även kunna analysera eventuella trender mellan dessa ytterligheter inkluderades även en mellanliggande segmentstorlek på 4000 tokens.

### 3.2.4 Overlap

Bennani och Moslonka [19] visar att overlap mellan segment inte ger någon mätbar förbättring i retrieval-prestanda och samtidigt ökar lagrings- och beräkningskostnader. I deras studie används däremot relativt små segment (50–500 tokens). Detta arbete använder betydligt större segment på 1000–8000 tokens vilket medför att resultaten därav inte nödvändigtvis är direkt överförbara på denna studie. Tidigare forskning har också pekat på att segmentering utan hänsyn till semantisk struktur kan leda till att relaterad information delas upp mellan flera segment, vilket kan resultera i fragmenterad kontext [16]. Detta arbete använder fixed-size chunking, det vill säga segmentering utan hänsyn till semantisk struktur. För att motverka att relevant information delas upp mellan segmentens gränser används en overlap på 50 tokens för det utvecklade QA-systemet. Rekommendationer för RAG-system föreslår ofta större overlap-värden, vanligtvis omkring 10-25% av segmentstorleken, för att bevara kontext mellan segment [13]. Det utvecklade QA-systemet i detta arbete är däremot inte ett traditionellt RAG-system, vilket innebär att dessa rekommendationer inte nödvändigtvis är direkt överförbara. Eftersom studien inte syftar till att optimera overlap-parametern, utan undersöka hur val av PDF-läsare, segmentstorlek samt sammanfattningslängd påverkar systemets prestanda, används samma overlap-värde i samtliga experiment.

### 3.2.5 Sammanfattningar

Sammanfattningen är en central del i det utvecklade QA-systemet då det innebär en komprimering av den långa kontexten till en mer hanterbar mängd kontext inför det frågebesvarande steget. Denna del innefattar flera viktiga designval för hur själva sammanfattningen konstrueras. I denna studie så genomförs sammanfattningen med en kombination av hierarkisk sammanfattning och frågefokuserad sammanfattning. Användningen av hierarkisk sammanfattning motiveras av att metoden visats vara särskilt effektiv på långa dokument som riskerar att överstiga språkmodellens totala kontextfönster [33]. Samtidigt kan metoden innebära en risk för informationsförlust vid fel mellan de olika hierarkiska nivåerna. Ou et al. [33] visar att fel vid tidiga sammanfattningssteg kan ha stor påverkan på senare steg, vilket påverkar den slutliga kvaliteten negativt. Detta motiverar valet av att komplettera hierarkiska

sammanfattningarna med ett frågefokuserat upplägg. Alanzi et al. [30] lyfter fram att frågefokuserad sammanfattning är särskilt användbar för att producera sammanfattningar där information som är särskilt intressant för aktuella frågeställningen prioriteras. Detta är relevant i det utvecklade QA-systemet då syftet med sammanfattningssteget är att skapa en så relevant global kontext som möjligt inför det slutgiltiga frågebesvaringssteget. Vid generering av sammanfattningarna inkluderas därför studiens frågor i prompten. Ett alternativt upplägg hade varit att tillämpa hierarkisk och frågefokuserad sammanfattning enskilt. Däremot bedömdes metoderna lämpliga att kombinera eftersom de adresserar olika utmaningar i pipeline-strukturen, dels genom att reducera informationsmängden, dels genom att styra sammanfattningarna mot relevant kontext. För att kunna jämföra olika varianter av den valda sammanfattningsstrategin varierades längden på sammanfattningarna mellan experimenten. Detta gjordes även mot bakgrund av tidigare forskning som visar att mängden text som inkluderas i en språkmodells kontext kan påverka modellens prestanda, särskilt vid identifiering av information i mitten av kontexten [1]. Efter initial testning så noterades att kontexten efter det första sammanfattningssteget sällan översteg 2000 tokens. Därför så testas två olika maxlängder på sammanfattning, en kortare om max 750 tokens och en längre om max 1500 tokens. Sammanfattningar genereras med hjälp av den valda språkmodellen *mistral-small3.2:24b*.

### 3.3 Dataset

Datasetet som används i studien är en samling PDF-dokument inom finansdomänen. Dessa består av fyra olika typer av dokument: årsredovisningar, prospekt, fondbestämmelser och fondgodkännande. Rapportförfattarna har även fått tillgång till en JSON-fil innehållande utvalda informationsfält från respektive dokument i form av *key-value*-par, där varje nyckel representerar ett specifikt fält och där varje värde utgör ett referenssvar. Varje dokument i JSON-filen har tillhörande sex fält:

- FundManagerName
- FundName
- SubFunds
- UmbrellaName
- DirectParent
- UltimateParent

Värdet består av antingen specifika namn eller tomma värden om information för specifikt fält saknas i ett visst dokument.

Datasetet består av:

- 26 dokument
- JSON-fil med fält och tillhörande korrekta svar för varje dokument

PDF:erna varierar i storlek, mellan 6 sidor och 175 sidor. Utöver detta varierar PDF:erna i språk, däribland svenska, engelska, danska, norska, finska, franska, spanska och isländska.

Experimenten genomförs på samtliga ovan nämnda PDF-dokument. Utifrån de fält som beskrivs ovan har motsvarande frågor formulerats:

- What is the fund manager name?
- What is the fund name?
- What is the umbrella fund name?
- What are the sub-fund(s) names?
- What is the direct parent?
- What is the ultimate parent?

Frågorna används i prompten både vid sammanfattning och i det efterföljande QA-steget. De svar som genereras i QA-steget jämförs därefter mot värdet i respektive frågas tillhörande fält vid utvärdering.

Ett exempel på annoterad data visas i figur 3.1. För dokumentet *fund\_doc\_Irish* ställs frågan ”What are the sub-fund(s) names?”. I dokumentutdraget identifieras tre sub-funds, vilket motsvarar värdena i SubFunds-fältet i den tillhörande JSON-filen.

1. The initial sub-funds of Nordea ETF ICAV:

BetaPlus Enhanced European Select Equity UCITS ETF

BetaPlus Enhanced Global Sustainable Equity UCITS ETF

BetaPlus Enhanced Global Developed Sustainable Equity UCITS ETF

are approved. The prior approval of the Central Bank is required if Nordea ETF ICAV intends to establish further sub-funds.

```
"SubFunds": [
  "BetaPlus Enhanced European Select Equity UCITS ETF",
  "BetaPlus Enhanced Global Sustainable Equity UCITS ETF",
  "BetaPlus Enhanced Global Developed Sustainable Equity UCITS ETF"
]
```

Figur 3.1: Urklipp ur dokumentet *fund\_doc\_Irish* med tillhörande facit från JSON-filen.

### 3.4 Experimentupplägg

För att utvärdera den föreslagna metoden har ett experimentupplägg utformats bestående av två faser, vilka beskrivs nedan.

#### 3.4.1 Fas 1 – Parameterstudie

Syftet med fas 1 är att undersöka hur olika designval påverkar prestandan hos det utvecklade QA-systemet. Parametrarna kombineras systematiskt, där en parameter ändras åt gången. Varje konfiguration körs två gånger på samtliga frågor mot samtliga dokument, och resultaten lagras för vidare analys. Målet är att identifiera en stark konfiguration för vidare jämförelse i fas 2.

Parametrar som varierar är metod för textutvinning (PyPDF och PyMuPDF4LLM), segmentstorlek (1000, 4000 och 8000 tokens) och sammanfattningslängd (750 och 1500 tokens). Parametrar som behövs för experimentet men som hålls konstanta är overlap (50 tokens), språkmodell mistral-small3.2:24b samt prompts.

### 3.4.2 Fas 2 – Jämförelse mot RAG

I den andra fasen jämförs den konfiguration som uppvisade högst resultat i parameterstudien mot en RAG-pipeline. För att minska påverkan av slumpmässig variation genomförs tre oberoende körningar av både den bäst presterande QA-konfigurationen som hittats samt RAG-pipelinan. Syftet är att sätta den föreslagna metoden i ett sammanhang och undersöka hur väl den presterar i förhållande till en redan etablerad metod för dokumentbaserad frågebesvarning. För att göra jämförelsen rättvis används samma språkmodell (*mistral-small3.2:24b*) och samma strukturerade PDF-läsare (*PyMuPDF4LLM*) i RAG-pipelinan som i det utvecklade QA-systemet i fas 1. Utöver detta definierades ett antal RAG-specifika parametrar baserat på tidigare forskning och etablerade riktlinjer.

Segmentstorleken sattes till 128 tokens, baserat på Bhat et al. [23], vilka visade att mindre segmentstorlekar, mellan 64–128 tokens presterar väl för faktabaserade frågor med korta tydliga svar. Då de svar som eftersöktes i denna studie huvudsakligen bestod av korta faktabaserade svar, såsom fondnamn, fondbörvaltare och moderbolag, bedömdes en segmentstorlek på 128 tokens vara lämplig i RAG-systemet.

För att minska risken att relevant information delas upp mellan segment användes overlap. Då segmentstorleken sattes till 128 tokens sattes overlap till 32 tokens, motsvarande cirka 25 % av segmentstorleken. Valen baserades på rekommendationer och standardvärden från Microsoft och NVIDIA, där overlap omkring 10-25% av segmentstorleken nämns [24,25].

Retrievaldjupet sattes till top-k = 5, vilket innebär att de fem högst rankade segmenten från vektordatabasen skickades vidare till språkmodellen efter semantisk sökning. Bhat et al. använde top-k-värden mellan 1 och 5 vid utvärdering av retrieval-prestanda, vilket motiverade valet av 5 som ett rimligt retrievaldjup för denna studie.

Similarity score-threshold valdes till 0.55, vilket baserades på forskning av Radeva et al. [26], där olika threshold-värden utvärderades inom RAG-system. Studien visade att en modell inom Mistral-familjen (*Mistral7b*) uppnådde bäst prestanda vid ett threshold-värde på 0.55, vilket motiverade valet av att använda samma threshold värde för den valda språkmodellen, *mistral-small3.2:24b*, i denna studie.

För embeddinggenerering användes modellen *multilingual-e5-base*, en flerspråkig embeddingmodell från Microsofts E5-familj, utvecklad för semantisk informationsretrieval. Modellen bygger på *xlm-roberta-base* och stödjer 100 språk [22]. Valet motiverades av att datasetet innehåller dokumentation på flera olika språk, vilket gjorde en flerspråkig embeddingmodell lämplig för experimentet.

## 3.5 Prompts

Det utvecklade QA-systemet styrs genom instruktioner till språkmodellen i samtliga steg. Totalt används tre olika prompttyper, där varje prompt är anpassad för ett specifikt steg i processen.

Först används en frågefokuserad extraktionsprompt där samtliga frågor skickas tillsammans med varje segment. Modellen instrueras att identifiera och extrahera information som kan vara relevant för de aktuella frågorna. Därefter kombineras och

bearbetas resultaten genom en reduktionsprompt som successivt sammanfattar innehållet till en mindre och mer hanterbar representation. När kontexten reducerats så att den understiger den valda sammanfattningslängden används slutligen en QA-prompt för att generera det slutgiltiga svaret. Promptdesignen har inspirerats av etablerade riktlinjer för prompt engineering från OpenAI och Microsoft [8,12,13]. Rekommendationerna innefattar bland annat tydliga instruktioner, begränsningar, strukturerade outputformat, uppdelning av komplexa uppgifter i mindre delsteg samt explicita instruktioner för hur modellen ska agera vid osäkerhet. Modellen instrueras i samtliga fall att endast använda tillhandahållen kontext, inte gissa vid osäkerhet samt returnera tomma svar om relevant information saknas. Samtliga prompts utformades enligt zero-shot-strategi. Eftersom studien hade tillgång till en relativt begränsad datamängd bedömdes zero-shot prompting vara lämpligt, då metoden inte kräver separat träningsdata [11]. Det övergripande arbetsflödet för systemet visas i figur 3.2.

```

chunks <- split(document)

Steg 1: Summering - Ett LLM-anrop per chunk

FOR chunk_i IN chunks:
    extractions[i] <- LLM("[frågor] [roll] [instruktion] [chunk_i] [begränsning] -> JSON")

Steg 2: Hierarkisk summering - komprimera tills vald sammanfattningslängd nåtts

WHILE tokens(extractions) > threshold:
    extractions <- LLM("[roll] [instruktion] [extractions] -> JSON")

global_context <- extractions

Steg 3: QA - ett sista LLM-anrop

answers <- LLM ("[frågor] [roll] [instruktion] [global_context] [begränsning] -> JSON")

RETURN answers

```

Figur 3.2: Pseudokod för det utvecklade QA-systemet.

Till skillnad från det utvecklade QA-systemet, där flera separata promptsteg används, använder RAG-pipelinan endast en QA-prompt. De två promptsteg som det utvecklade QA-systemet använder för att identifiera relevant information ersätts i RAG-pipelinan av en semantisk sökning i en vektordatabas innan QA-steget genomförs. QA-prompten i RAG-pipelinan utformas däremot enligt samma riktlinjer som det utvecklade QA-systemets avseende roller, begränsningar, agerande vid osäkerhet samt outputformat.

### 3.6 Genomförande

Experimenten genomförs som batchkörningar där varje konfiguration testas över samma uppsättning dokument och frågor. Resultaten lagras i ett strukturerat JSON-format för att möjliggöra jämförelse mellan olika konfigurationer. Totalt testas 12 olika konfigurationer i parameterstudien. Varje konfiguration körs två gånger för att öka resultatens tillförlitlighet.

## Konfigurationstabell

Tabell 3.1: Testade konfigurationer för det utvecklade QA-systemets parameterstudie.

Konfiguration	Segmentstorlek	Sammanfattningslängd	PDF-läsare
1	1000	750	PyMuPDF4LLM
2	1000	750	PyPDF
3	1000	1500	PyMuPDF4LLM
4	1000	1500	PyPDF
5	4000	750	PyMuPDF4LLM
6	4000	750	PyPDF
7	4000	1500	PyMuPDF4LLM
8	4000	1500	PyPDF
9	8000	750	PyMuPDF4LLM
10	8000	750	PyPDF
11	8000	1500	PyMuPDF4LLM
12	8000	1500	PyPDF

### 3.7 Utvärdering

I denna studie används Exact Match (EM) och F1-värde för utvärdering, då dessa är etablerade mått inom frågebesvarande (QA) och bland annat används i SQuAD-datasetet introducerat av Rajpurkar et al. [39].

Resultaten utvärderas genom jämförelse med ett fördefinierat facit.

- Exact Match - mäter andelen svar som exakt matchar referenssvaret
- F1-värde - mäter tokenbaserad överlappning mellan predikterat svar och referenssvaret

Dessa mått används för att jämföra både olika konfigurationer för det utvecklade QA-systemet i fas 1 och för QA-systemet mot RAG-metoden i fas 2. Genom att använda båda måtten möjliggörs en analys av såväl strikt korrekthet som delvis kor-

rekta svar. ROUGE, som är vanligt förekommande mått vid utvärdering av textsammanfattningar, används inte i denna studie då fokus ligger på att utvärdera korta, faktabaserade svar för systemen i stort snarare än att utvärdera sammanfattningarnas kvalitet i sig. Mänsklig utvärdering hade kunnat användas som komplement till de automatiska måtten, men eftersom studien i huvudsak består av korta faktabaserade värden bedömdes EM och F1-värde vara tillräckliga för studiens syfte samt möjliggöra reproducerbara automatiserade jämförelser.

Innan jämförelse normaliseras svaren genom omvandling till gemener samt borttagning av skiljetecken. För fält som innehåller flera värden, såsom *SubFunds*, jämförs mängden predikterade värden mot motsvarande mängd i facit utan hänsyn till ordning. Om modellen returnerar extra eller saknar värden ges EM = 0 för det aktuella fältet. Exact Match beräknas därmed på hela fältet som en enhet. Tomma prediktioner räknas som korrekta om motsvarande facit också är tomt. F1-värde inkluderas för att även fånga delvis korrekta svar och mindre avvikelser som inte fångas av Exact Match. För att illustrera hur Exact Match och F1-värde påverkas av olika typer av prediktioner visas exempel i tabell 3.2. I resultatkapitlet presenteras Exact Match i vissa fall även tillsammans med antal helt korrekta prediktioner för att underlätta tolkningen av resultaten.

Tabell 3.2: Exempel på hur olika predikterade svar kan påverka Exact Match och F1-värde.

Predikterat svar	Facit	EM	F1
CLO OPPORTUNITY FUND	CLO OPPORTUNITY FUND	1,00	1,00
Ardesia SCA SICAV-RAIF-CLO OPPORTUNITY FUND	CLO OPPORTUNITY FUND	0,00	0,67
Fund A; Fund B, Fund C	Fund A; Fund B	0,00	0,80
Moneda Deuda Latinoamericana Fondo de Inversión	Moneda Deuda Lationamericana Fondo de Inversion	0,00	0,83



## 4 Resultat

I detta kapitel presenteras studiens resultat. Först redovisas parameterstudien för det utvecklade QA-systemet, där tolv konfigurationer utvärderades genom två körningar vardera. Därefter presenteras en jämförelse mellan den bästa identifierade konfigurationen och en tidigare utvecklad RAG-pipeline. I jämförelse mellan systemen genomfördes tre oberoende körningar per system, där medelvärden används vid resultatredovisning. Resultaten presenteras både som övergripande mått samt uppdelat per fält och dokumenttyp.

### 4.1 Utvärderingsupplägg

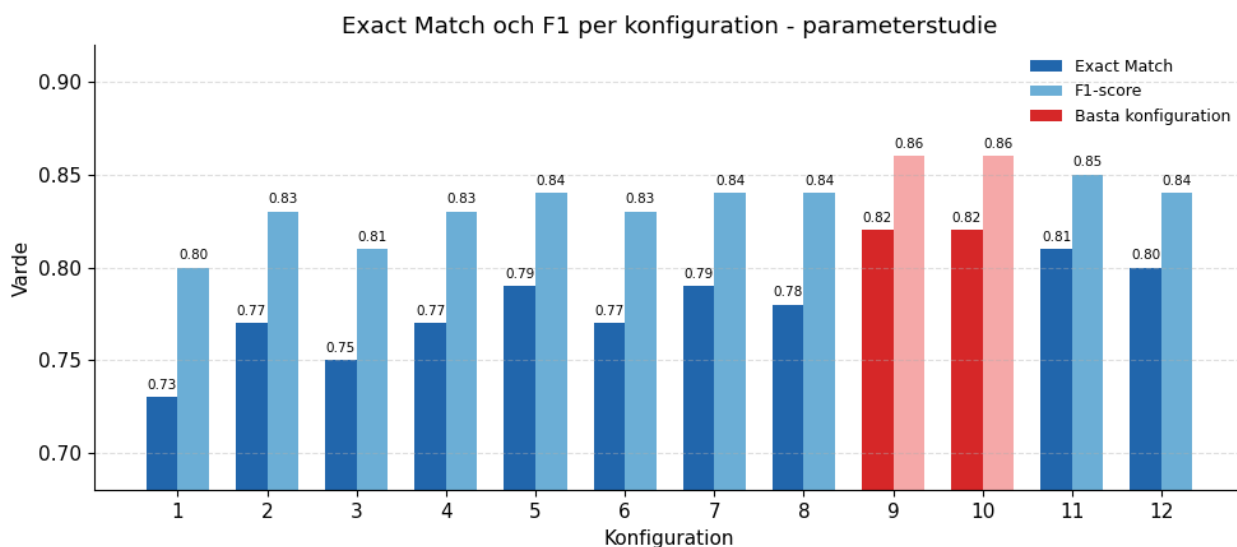
För att möjliggöra systematisk utvärdering implementerades stöd för batchkörningar, där olika parameterkombinationer kunde testas automatiskt. När en körning med vald konfiguration slutförts, genererades automatiskt en mapp med tillhörande utvärderingsfil innehållande resultat för hur konfigurationen presterat på olika dokument och frågor. Här angavs värden för *Exact Match* och *F1-värde*, vilka därefter kunde jämföras mellan systemets olika konfigurationer. Även RAG-pipelinen hade motsvarande stöd för batchkörning och resultatgenerering implementerad, vilket möjliggjorde jämförelser mellan det utvecklade QA-systemets och RAG-pipelinens resultat.

### 4.2 Resultat av parameterstudie

Parameterstudien omfattade tolv konfigurationer, framtagna för kombination av tre segmentstorlekar (1000, 4000 och 8000 tokens), två sammanfattningslängder (750 och 1500 tokens) samt två PDF-läsare (PyPDF och PyMuPDF4LLM). Overlap på 50 tokens användes för samtliga konfigurationer. Varje konfiguration kördes två gånger och utvärderades på samtliga 26 dokument. Resultaten redovisas i tabell 4.1 och figur 4.1.

Tabell 4.1: Exact Match och F1 för samtliga tolv konfigurationer i parametersökningen. Segmentstorlek och sammanfattningslängd är angivet i antal tokens. Värden representerar medelvärdet av två oberoende körningar. Diff anger absolutbeloppet av skillnaden i EM respektive F1 mellan de två körningarna. Bästa konfigurationerna som hittades är markerade med fetstil.

Konfiguration	Segmentstorlek	Sammanfattningslängd	PDF-läsare	Medel EM	Diff EM	Medel F1	Diff F1
1	1000	750	PyMuPDF4LLM	0,73	0,01	0,80	0,01
2	1000	750	PyPDF	0,77	0,02	0,83	0,01
3	1000	1500	PyMuPDF4LLM	0,75	0,02	0,81	0,01
4	1000	1500	PyPDF	0,77	0,02	0,83	0,01
5	4000	750	PyMuPDF4LLM	0,79	0,02	0,84	0,02
6	4000	750	PyPDF	0,77	0,03	0,83	0,02
7	4000	1500	PyMuPDF4LLM	0,79	0,01	0,84	0,01
8	4000	1500	PyPDF	0,78	0,05	0,84	0,04
<b>9</b>	<b>8000</b>	<b>750</b>	<b>PyMuPDF4LLM</b>	<b>0,82</b>	<b>0,01</b>	<b>0,86</b>	<b>0,00</b>
<b>10</b>	<b>8000</b>	<b>750</b>	<b>PyPDF</b>	<b>0,82</b>	<b>0,02</b>	<b>0,86</b>	<b>0,01</b>
11	8000	1500	PyMuPDF4LLM	0,81	0,02	0,85	0,02
12	8000	1500	PyPDF	0,80	0,01	0,84	0,01



Figur 4.1: Exact Match och F1 för samtliga tolv konfigurationer i parametersökningen. Värden representerar medelvärdet av två oberoende körningar. Bästa konfigurationerna (9 och 10) är markerade i rött.

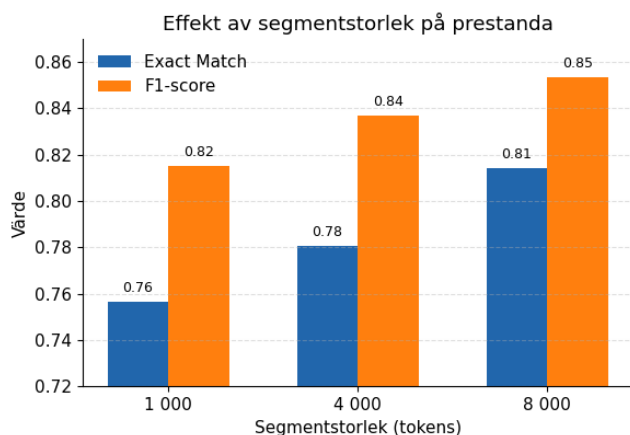
### 4.2.1 Resultat per parameter

För att undersöka hur olika parametrarna påverkade systemets prestanda genomfördes en separat analys av segmentstorlek, sammanfattningslängd och PDF-läsare. Parameteranalysen baseras på ett medelvärde av resultaten där varje konfiguration kördes två gånger.

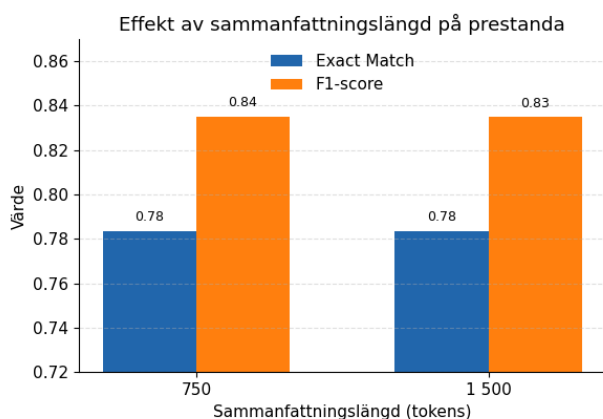
Figur 4.2 visar segmentstorlekens påverkan på resultatet. Inom det testade intervallet observerades generellt högre Exact Match och F1-värden för större segmentstorlekar, där 8000 tokens gav högst genomsnittligt resultat. Skillnader mellan de olika testade segmentstorlekarna bedöms vara tydligare än skillnaderna mellan de testade sammanfattningslängderna och PDF-läsarna.

Figur 4.3 visar sammanfattningslängdens påverkan på resultatet. Resultaten indikerar att sammanfattningslängden hade en relativt begränsad påverkan på systemets prestanda, även om 750 tokens tenderade att ge något högre eller likvärdiga resultat jämfört med 1500 tokens. Skillnaden mellan olika sammanfattningslängder bedöms vara små.

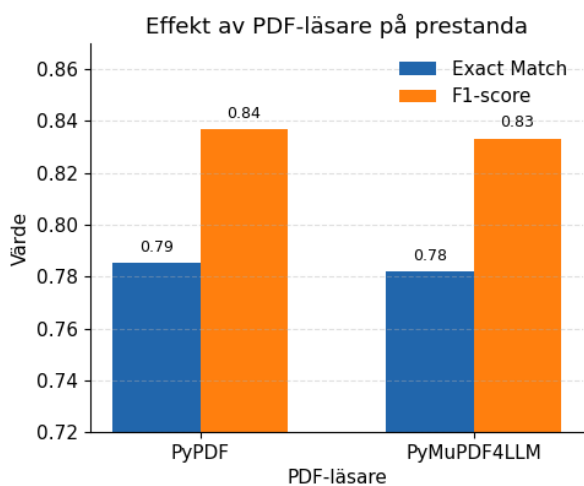
Figur 4.4 visar skillnader mellan de olika PDF-läsarna. Resultaten för PyPDF och PyMuPDF4LLM var övergripande likvärdiga, där mindre skillnader observerades mellan de två experimentomgångarna. I en av körningarna presterade PyMuPDF4LLM något bättre, medan PyPDF uppvisade något högre värde i den andra körningen. Skillnaderna mellan PDF-läsarna bedöms vara små.



Figur 4.2: Visar genomsnittligt EM och F1-värde vid varierande segmentstorlek för det utvecklade QA-systemet över två oberoende körningar.



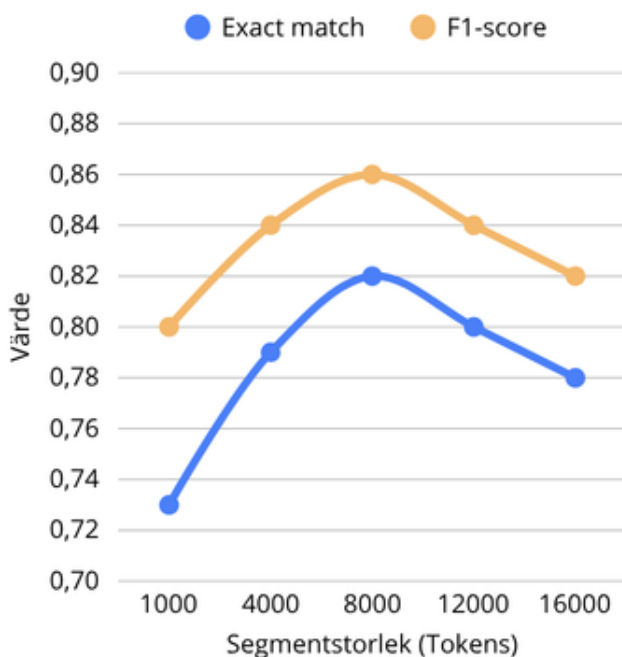
Figur 4.3: Visar genomsnittligt EM och F1-värde vid varierande sammanfattningslängd för det utvecklade QA-systemet över två oberoende körningar.



Figur 4.4: Visar genomsnittligt EM och F1-värde vid varierade testade PDF-läsare för det utvecklade QA-systemet över två oberoende körningar.

#### 4.2.2 Kompletterande analys av segmentstorlek

Då trenden för segmentstorlek i parameterstudien såg ut att indikera att högre segmentstorlek gynnar QA-systemet, testades också två högre segmentstorlekar: 12000 och 16000 tokens. Detta gjordes för att undersöka om det finns en högre segmentstorlek som är optimal men som inte inkluderades i parameterstudien. Två körningar för respektive segmentstorlek testades, där PDF-läsaren PyMuPDF4LLM valdes och sammanfattningslängden valdes till 750 tokens. Resultaten visas i figur 4.5. Kurvan indikerar att den segmentstorlek som noterats som bästa tidigare (8000 tokens), presterar bättre än högre segmentstorlekar.



Figur 4.5: Visar genomsnittligt EM och F1-värde för varierade segmentstorlekar, där PyMuPDF4LLM och sammanfattningslängd på 750 tokens används.

### 4.3 Jämförelse – QA-system vs RAG

För att jämföra det utvecklade QA-systemet mot RAG-pipelin utvärderades den bästa konfigurationen som hittades från parameterstudien mot motsvarande RAG-konfiguration. Två konfigurationer (konf. 9 och 10) identifierades som likvärdiga i parameterstudien. För vidare jämförelse mot RAG-pipelin valdes konfiguration 9, vilken använde en segmentstorlek på 8000 tokens, sammanfattningslängd på 750 tokens samt PDF-läsaren PyMuPDF4LLM. Konfigurationen uppnådde högst resultat i den initiala experimentomgången och uppvisade samtidigt låg variation mellan de två genomförda körningarna.

För att undersöka variation mellan körningar genomfördes tre oberoende körningar per system. Resultaten utvärderades med hjälp av Exact Match och F1-värde på övergripande nivå samt per fält.

#### 4.3.1 Overall-resultat – QA-system vs RAG

Resultaten per körning samt medelvärde redovisas i tabell 4.2. För QA-systemet inkluderades den initiala körningen från parameterstudien tillsammans med två ytterligare oberoende körningar. Variationen mellan utförda körningar uppgår till +/- 0,01 EM-enheter för båda systemen.

Tabell 4.2: Exact Match och F1-värde för tre oberoende körningar av bästa konfiguration som hittats i parameterstudien (konfiguration 9) samt RAG-systemet, med medelvärde per system. Kolumnen X/156 anger helt korrekta prediktioner baserat på respektive systems medelvärde, i relation till totalt 156 frågor (26 dokument x 6 fält).

System	Körning 1	Körning 2	Körning 3	Medel	X/156
QA-system EM	0,82	0,81	0,81	0,81	127/156
QA-system F1	0,86	0,85	0,85	0,85	-
RAG EM	0,77	0,76	0,76	0,76	119/156
RAG F1	0,81	0,81	0,81	0,81	-

#### 4.3.2 Resultat per fält/fråga

För att analysera vilka fält som systemen hade lättast respektive svårast att prediktera presenteras resultaten per fält i tabell 4.3, 4.4 samt figur 4.6. Utöver genomsnittliga EM och F1-värden redovisas även i tabell 4.3 hur många av de 26 dokumenten där respektive fält predikterades helt korrekt för systemen.

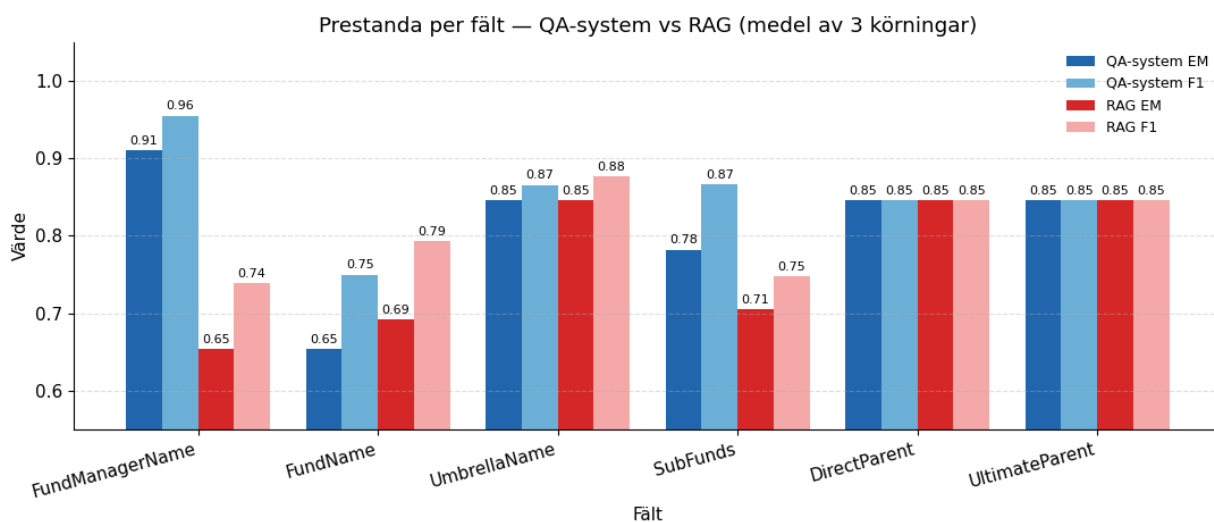
Resultaten visar att QA-systemet uppnådde högre resultat för FundManagerName och SubFunds, där även de största skillnaderna mellan systemen observerades. För FundName och UmbrellaFundName uppnådde RAG-systemet något högre resultat, medan systemen presterade identiskt när det gäller DirectParent och UltimateParent.

Tabell 4.3: Exact Match per fält, baserat på medelvärdet över tre körningar. Kolumnen X/26 anger antalet helt korrekta prediktioner per fält i relation till studiens 26 dokument.

Fält	QA-system EM	QA-system X/26	RAG EM	RAG X/26
FundManagerName	0,91	24/26	0,65	17/26
FundName	0,65	17/26	0,69	18/26
UmbrellaFundName	0,85	22/26	0,85	22/26
SubFunds	0,78	20/26	0,71	18/26
DirectParent	0,85	22/26	0,85	22/26
UltimateParent	0,85	22/26	0,85	22/26

Tabell 4.4: F1-värde per fält, medelvärde över tre körningar. Även differens mellan systemens F1-värde.

Fält	QA-system F1	RAG F1	Differens
FundManagerName	0,96	0,74	+0,22 QA-system
FundName	0,75	0,79	+0,04 RAG
UmbrellaFundName	0,87	0,88	+0,01 RAG
SubFunds	0,87	0,75	+0,12 QA-system
DirectParent	0,85	0,85	lika
UltimateParent	0,85	0,85	lika



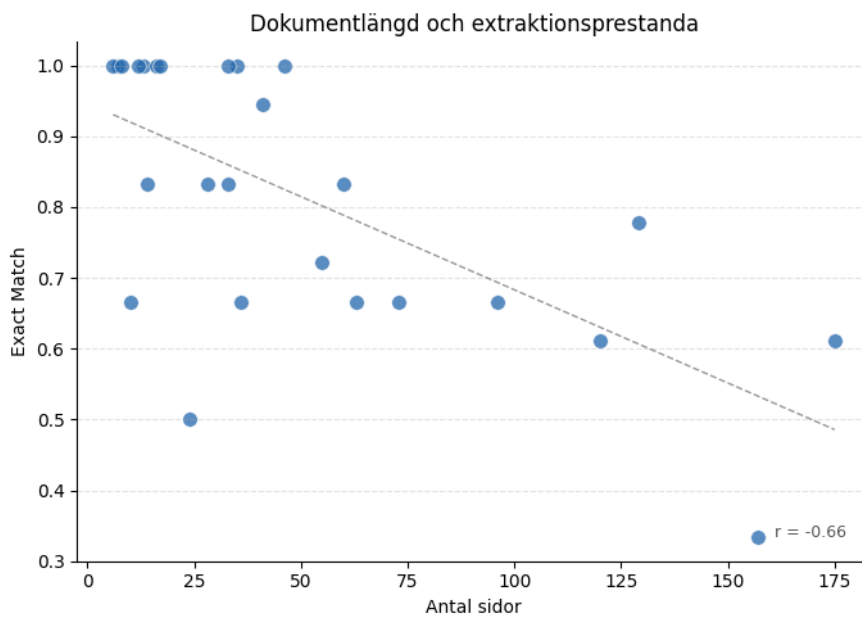
Figur 4.6: Visar EM och F1-värde för samtliga fält för konfiguration 9 samt för RAG, medelvärde över tre körningar.

#### 4.4 Dokumentkomplexitet

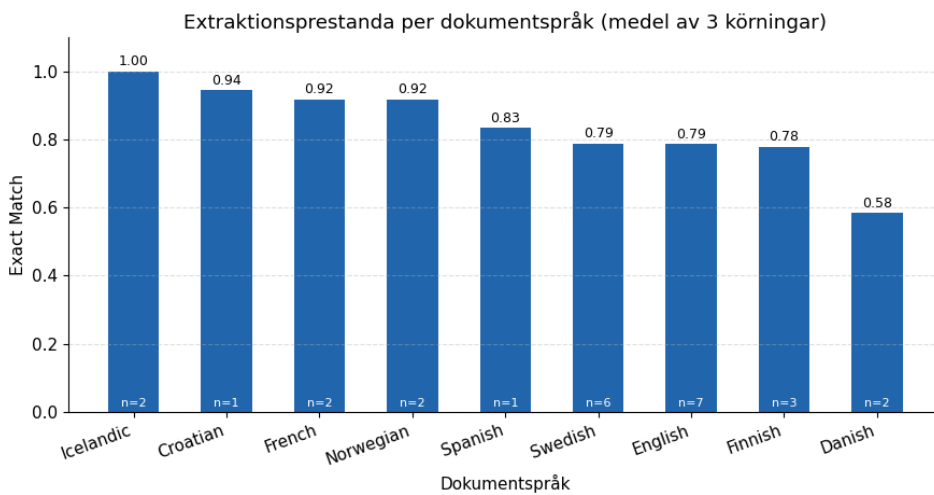
Dokumenterna som ingick i studien varierade både i längd och språk, vilket potentiellt kan påverka systemets extraktionsförmåga. För att undersöka detta analyserades sambandet mellan dokumentlängd och Exact Match (Se figur 4.7) samt mellan dokumentets språk och systemets prestanda (Se figur 4.8) för det utvecklade QA-systemet.

Figur 4.7 indikerar en möjlig trend där längre dokument generellt uppvisade lägre resultat.

Figur 4.8 indikerar viss variation i prestanda mellan språk. Danska sticker ut något med ett EM-värde på 0.58.



Figur 4.7: Visar samband mellan EM och dokumentlängd för det utvecklade QA-systemet, medelvärde över tre körningar.



Figur 4.8: Visar samband mellan EM och språk, sorterat fallande där n innebär antal dokument per språk. Språk med  $n = 1$  bör tolkas med försiktighet.

## 5 Analys och diskussion

Syftet med studien var att undersöka hur en flerstegspipeline för frågebesvarande påverkar QA-prestandan, samt att jämföra metoden mot en RAG-baserad lösning. För att undersöka detta utvecklades ett system där dokument successivt bearbetades genom dataextrahering, textsegmentering, sammanfattning och slutgiltigt frågebesvarande.

Analysen behandlar hur olika designval för systemet påverkade resultaten, hur den föreslagna metoden förhåller sig till resultaten för RAG-systemet, samt vilka begränsningar som finns i experimentupplägg och utvärderingsmått. Vidare diskuteras även hållbarhetsaspekter kopplat till det utvecklade systemet i relation till ett RAG-system.

### 5.1 Analys av parameterstudien

Pipelinen testades i tolv konfigurationer genom att kombinera tre segmentstorlekar (1000, 4000 och 8000 tokens), två sammanfattningslängder (750 och 1500 tokens) samt två PDF-läsare (PyPDF och PyMuPDF4LLM). Samtliga körningar kördes två gånger i den initiala experimentomgången och utvärderades på 26 dokument med 6 fält per dokument.

Resultaten indikerar att större segmentstorlek generellt gav bättre resultat inom det testade intervallet. Genomsnittliga F1-värdet för varje storleksnivå var 0,82 (1000 tokens), 0,84 (4000 tokens) och 0,85 (8000 tokens), vilket framgår av figur 4.2. Genomsnittliga EM-värdet för varje segmentstorleksnivå var 0,76 (1000 tokens), 0,78 (4000 tokens), 0,81 (8000 tokens). Resultaten indikerar att textsegment av större storlek bidrar positivt till det utvecklade systemet. En möjlig förklaring är att modellen ges mer sammanhängande text att arbeta med, vilket minskar risken att relevant kontext hamnar mellan olika segment och inte kopplas samman. Resultatet ligger även i linje med Ou et al. [33], som använde segmentstorlekar på upp till 8000 tokens vid hierarkisk sammanfattning av långa dokument. Trenden i resultaten motiverade en kompletterande undersökning av större segmentstorlekar än de som inkluderades i parameterstudien. Därför testades även segmentstorlekar på 12000 och 16000 tokens. Resultaten från dessa tester visade däremot lägre prestanda än för 8000 tokens, vilket tyder på att den optimala segmentstorleken uppnåddes vid 8000 tokens, vilket framgår i figur 4.5.

Utifrån tidigare forskning kring *lost in the middle*-problemet hade mindre segmentstorlekar kunnat förväntas ge bättre resultat, då längre kontexter kan försvåra språkmodellens förmåga att identifiera relevant information [1]. Trots detta observerades i studien att större segmentstorlekar förbättrade prestandan upp till 8000 tokens, medan ytterligare öknings gav sämre resultat. En möjlig förklaring är att bevarandet av mer sammanhängande kontext vägde tyngre än de nackdelar som längre textsegment potentiellt medförde. Experimenten baserades på fixed-size chunking, vilket kan innebära att semantiskt relaterad information delas upp mellan flera textsegment, då segmenteringen inte tar hänsyn till dokumentets semantiska struktur

[16]. Ett alternativt angreppssätt hade därför varit att använda andra segmenteringsstrategier, såsom semantic chunking eller proposition-based chunking, vilket tidigare forskning visat kan förbättra retrieval-kvaliteten jämfört med fixed-size chunking [17].

Sammanfattningslängden uppvisade ingen tydlig inverkan på QA-systemets prestanda, även om resultaten i genomsnitt indikerade att 750 tokens presterade något bättre än 1500. Skillnaderna mellan konfigurationerna var relativt små, och varierade något mellan körningarna, vilket gör det svårt att dra några starka slutsatser kring den optimala sammanfattningslängden. En möjlig förklaring är att den angivna sammanfattningslängden endast fungerade som ett övre tak snarare än en fast längd. Eftersom dokumenten i datasetet varierade kraftigt i storlek, från ett fåtal sidor till betydligt större dokument, blev många sammanfattningar naturligt kortare än den maximalt tillåtna längden, vilket sannolikt minskade den praktiska skillnaden mellan konfigurationerna. Variationen kan även ha påverkats av promptens utformning, då modellen instruerades att generera strikt strukturerad JSON-output, vilket sannolikt bidrog till mer komprimerade sammanfattningar. Sammanfattningarnas innehåll och detaljinnehåll kan därför ha påverkats av promptdesignen, även om detta inte undersöktes specifikt i denna studie. Eftersom hierarkisk och frågefokuserad sammanfattning användes i kombination är det svårt att avgöra vilken av metoderna som hade störst inverkan på resultaten. Ett alternativt experimentuppställning hade varit att utvärdera metoderna separat för att tydligare isolera respektive methods effekt på QA-prestandan, i stället för att huvudsakligen fokusera på hur sammanfattningslängden påverkade resultaten.

PDF-läsarens inverkan i experimenten var begränsad. För vissa konfigurationer presterade PyMuPDF4LLM något bättre, medan PyPDF presterade bättre för andra. Sammantaget var skillnaderna mellan PDF-läsarna små, vilket indikerar att valet av de testade PDF-läsarna har relativt liten inverkan på systemets prestanda. Resultaten antyder att bevarandet av viss struktur genom markdown inte gav någon direkt mätbar fördel i det utvecklade systemet. En möjlig förklaring är att den information som krävdes för frågebesvarandet i många fall kunde identifieras även i den ostrukturerade textrepresentationen som genererades av PyPDF. Under experimenten observerades att viss dokumentstruktur fortfarande gick förlorad vid extraheringen från PDF-format till text, där exempelvis skillnader mellan huvudrubriker och underrubriker inte bibehölls i den extraherade texten, samt att tabeller i viss mån föll samman. En mer avancerad parser hade potentiellt kunnat förstärka skillnaderna mellan strukturerad och ostrukturerad textextraktion. Detta vore särskilt relevant i finansiella dokument, där information ofta förekommer både i form av tabeller och löpande text, vilket forskning identifierat som en utmaning vid textutvinning [3].

Bland de ursprungliga 12 konfigurationerna uppnådde konfiguration 9 och 10 högst resultat, med ett EM-värde på 0,82 och ett F1-värde på 0,86 i genomsnitt över de två körningarna. Den lägst presterande konfigurationen var konfiguration 1 med ett EM-värde på 0,73 och F1-värde på 0,80 i genomsnitt. Detta motsvarar en skillnad på 9 procentenheter för EM och 6 procentenheter för F1 mellan de högst och lägst presterande konfigurationerna. Resultaten indikerar att valet av konfiguration hade tydlig

påverkan på systemets prestanda, där segmentstorleken framstod som den parameter som hade störst inverkan.

## 5.2 Jämförelse mot RAG-baseline

Jämförelsen mellan systemen baserades på medelvärdet av tre oberoende körningar per system. För det utvecklade QA-systemet valdes en av de två bäst presterande konfigurationerna från parameterstudien ut för vidare jämförelse mot en RAG-pipeline. Den valda konfigurationen, konfiguration 9, uppnådde ett genomsnittligt EM-värde på 0,81 och ett genomsnittligt F1-värde på 0,85 över de tre körningarna.

Den använda RAG-konfigurationen (segmentstorlek=128, overlap=32, top-k=5, threshold=0.55) uppnådde ett genomsnittligt EM-värde på 0,76 och ett genomsnittligt F1-värde på 0,81, vilket framgår i tabell 4.2. Det utvecklade QA-systemets konfiguration 9 överträffade därmed RAG-systemet med 5 procentenheter för EM och 4 procentenheter för F1. Uttryckt i antal helt korrekta prediktioner motsvarade detta 127 korrekt besvarande fält av totalt 156 för QA-systemet, jämfört med 119 för RAG-pipelinen. Eftersom flera av fälten uppvisade endast marginella skillnader fokuserar den fortsatta analysen främst på de områden där tydligare skillnader mellan systemen kunde observeras.

Den mest markanta skillnaden återfinns för fältet *FundManagerName*, där det utvecklade QA-systemet uppnår EM=0,91 jämfört med RAG-pipelinens EM=0,65, vilket illustreras i tabell 4.3. Detta innebär en differens på 26 procentenheter. Uttryckt i antal helt korrekta prediktioner motsvarar detta 24 korrekt predikterade dokument av 26 för QA-systemet, jämfört med 17 av 26 för RAG-systemet. Den observerade skillnaden mellan de två systemen för detta fält kan bero på att information gällande *FundManagerName* vid granskning uttrycks på flera olika sätt i dokumenten. I dokumenten användes, i flertalet fall, alternativa formuleringar såsom *management company* i stället för *fund manager*. Detta kan ha påverkat retrieval-steget för RAG-pipelinen negativt, då relevanta textsegment inte alltid identifierades och hämtades. Att relevanta textsegment inte alltid identifieras är även något som forskning visar kan vara en utmaning för retrieval-baserade system [2]. Den utvecklade pipelinen verkar i högre grad kunna hantera de olika formuleringarna, vilket kan indikera att användningen av en språkmodell redan vid första sammanfattningssteg (som ersätter RAG-systemets retrieval) bidrar till att information uttryckt på varierade sätt hanteras bättre.

Fältet *SubFunds* gynnade även det utvecklade QA-systemet (EM=0,78 jämfört med 0,71, och F1=0,87 jämfört med 0,75). Detta motsvarar 20 helt korrekta prediktioner av 26 dokument för QA-systemet och 18 av 26 för RAG-systemet. Den större skillnaden i F1-värde kan indikera att det utvecklade QA-systemet oftare lyckades identifiera delar av de korrekta sub-funds även när samtliga sub-funds för ett visst dokument inte predikterades exakt korrekt. I datasetet observerades att antalet sub-funds varierar mellan fonder och dokument, från inga alls till flera förekomster. Frasen sub-funds kunde dessutom förekomma på flera olika platser i dokumenten utan att de relevanta namnen nämndes i direkt anslutning. Detta kan ha gjort det svårare för retrieval-steget i RAG-systemet att identifiera de segment där relevanta namnen faktiskt förekom. Resultaten indikerar därmed att det utvecklade

QA-systemet i den valda konfigurationen hade lättare att hantera fält där antalet svar varierade mellan dokumenten och där relevant information kunde vara utspridd i texten. Samtidigt är det möjligt att andra retrieval-parametrar, exempelvis ett högre antal hämtade segment hade kunnat förbättra resultaten för RAG-systemet.

Sammanlaget indikerar resultaten att det utvecklade QA-systemet hade störst fördel där relevant information kunde förekomma utspritt i dokumenten eller uttryckas på varierade sätt. För mer väldefinierade fält med förutsägbar placering i dokumentet framstår skillnaderna mellan systemen däremot små, och i vissa fall presterade RAG-systemet marginellt bättre.

Utöver fördelen att det utvecklade QA-systemet uppnådde högre EM och F1-resultat jämfört med RAG-pipelinan som testades i denna studie, erbjuder arkitekturen även andra praktiska fördelar. Det utvecklade QA-systemet eliminerar det retrieval-steg som utgör en central del i RAG-baserade system. I en RAG-pipeline avgör en semantisk vektorsökning vilka textsegmenten som skickas till språkmodellen, vilket innebär att relevant information riskerar att aldrig nå modellen om sökningen misslyckas med att identifiera rätt segment [2]. Det utvecklade QA-systemet kringgår detta genom att i stället låta kontext samlas ihop genom sammanfattning av PDF-dokumentens information. Informationsförlust kan ske även i en sådan design, där forskning visat att informationsförlust kan ackumuleras, speciellt vid hierarkisk sammanfattning i flera nivåer, då varje sammanfattningssteg bygger vidare på tidigare genererad sammanfattning [32]. Däremot finns det en garanti på att en språkmodell fått möjlighet att ta del av samtlig information minst en gång, vilket inte kan garanteras i ett retrieval-baserat system.

Ytterligare en fördel är systemets flexibilitet. Eftersom det utvecklade QA-systemet styr kontextutvinning genom promptar, via naturligt språk, kan systemets beteende justeras av en användare utan teknisk kunskap om den underliggande infrastrukturen. En domänexpert kan exempelvis förfina instruktionerna för en specifik fråga eller en specifik dokumenttyp direkt i prompttexten, utan att behöva modifiera vektordatabas, embeddingmodell eller indexeringslogik. Detta skulle kunna sänka tröskeln för praktisk användning.

### 5.3 Analys av dokumentkomplexitet

Analysen av dokumentkomplexitet indikerar ett möjligt negativt samband mellan dokumentlängd och systemets prestanda. Som visas i figur 4.7 observerades en trend där längre dokument generellt uppvisade lägre EM-värden. En möjlig förklaring är att längre dokument innehåller större informationsmängd och mer varierad kontext, vilket kan ha försvårat både sammanfattning och frågebesvarande. Resultaten bör däremot tolkas med försiktighet eftersom antalet dokument är begränsat och majoriteten av dokumenten återfinns bland de kortare dokumentlängderna.

När det gäller dokumentspråk observerades, som figur 4.8 visar, viss variation i prestanda mellan språken. Samtidigt framstår systemets prestanda som relativt jämn för majoriteten av de undersökta språken. Eftersom flera språk endast representeras

av ett fåtal dokument bör resultaten även här tolkas med försiktighet, och det är därför svårt att dra generella slutsatser om språkets faktiska påverkan på systemets prestanda.

#### 5.4 Jämförelse mot externa resultat

Som kompletterande referenspunkt genomförde uppdragsgivaren en manuell utvärdering av tre externa LLM-system: ChatGPT, Le Chat och Gemini, på samma dokumentuppsättning (se tabell 2.1). Till skillnad från den automatiserade evalueringen av QA-systemet baserades dessa resultat på manuell bedömning, där mindre variationer i exempelvis stavning eller formatering accepterades som korrekta svar. Trots skillnaderna i utvärderingsmetod visar resultaten att det utvecklade QA-systemet presterar i nivå med kommersiella LLM-system. QA-systemets genomsnittliga EM-värde över tre körningar (0,81) låg relativt nära ChatGPT (0,88) och överträffade både Le Chat (0,78) och Gemini (0,59). Uttryckt i antal korrekt predikterade fält motsvarade detta 127 korrekta prediktioner för QA-systemet, jämfört med cirka 137 för ChatGPT, 122 för Le Chat och 92 för Gemini. Resultaten tyder därmed på att det utvecklade systemet presterar konkurrenskraftigt i jämförelse med de testade externa systemen på den aktuella dokumentuppsättningen.

#### 5.5 Begränsningar

Studien innehåller flera begränsningar kopplade till använda mått och experimentupplägg. Nedan diskuteras de mest centrala begränsningarna som kunnat identifieras för studien.

##### 5.5.1 Begränsningar i använda mått

Utvärderingen av det utvecklade QA-systemet och RAG-pipelinan baserades på Exact Match (EM) och F1-värde, vilka är vedertagna mått inom informationsextrahering. Dessa mått visades däremot ha flera konkreta begränsningar i det aktuella sammanhanget.

Den första begränsningen gäller namn på sub-funds. Modellen extraherar namn i den form de förekommer i dokumentet, exempelvis *”Ardesia SCA SICAV-RAIF – CLO OPPORTUNITY FUND”*, medan JSON-filen som använts för svarsvalidering i vissa fall endast innehåller den avslutande delen av namnet, i detta fall *”CLO OPPORTUNITY FUND”*. Samma mönster syns i andra dokument där modellen inkluderar prefix, vilket är korrekt enligt hur flera av dessa namn presenteras i PDF-dokumentet, men inte i facit. Detta medför att EM i dessa fall blir 0 och att F1-värdet kan reduceras kraftigt, trots att informationsextraheringen i praktiken är fullständig.

En annan begränsning handlar om specialtecken. I ett av dokumenten får fondnamnet EM=0,0 trots att det predikterade svaret och facit i praktiken är identiska. Modellen returnerar *”Moneda Deuda Latinoamericana Fondo de Inversión”* medan namnet i facit är *”Moneda Deuda Latinoamericana Fondo De Inversion”*. Bokstäverna *ó* och *o* behandlas som olika tecken, vilket påverkar både EM och F1 negativt. Fynden tyder på att de uppnådda EM och F1-värdena i viss mån underskattar systemets faktiska korrekthet.

Med dessa begränsningar i beaktning hade det varit värdefullt med ytterligare sätt att validera resultaten, där mänsklig utvärdering hade kunnat fungera som ett komplement till de automatiska måtten. Mänsklig utvärdering hade troligtvis i större utsträckning kunnat bedöma den semantiska korrektheten i modellens svar, även i fall där formuleringen skiljde sig från referenssvaret.

### 5.5.2 Begränsningar i experimentupplägg

Det utvecklade QA-systemet testades i 12 konfigurationer för att identifiera den bäst presterande konfigurationen, medan RAG-pipelinen kördes i en konfiguration baserad på etablerade riktlinjer och tidigare forskning. Det är möjligt att en systematisk sökning över exempelvis segmentstorlekar, overlap och top-k hade kunnat förbättrat RAG-pipelinens resultat och minskat gapet mot det utvecklade QA-systemet. Jämförelsen ska därför tolkas som det utvecklade QA-systemets bästa konfiguration mot ett välmotiverat RAG-utgångsläge, inte mot ett fullt optimerat RAG-system. En annan aspekt är att endast en språkmodell testades. Båda systemen kördes uteslutande med *mistral-small3.2:24b*. Därav säger resultaten något om hur denna modell presterar med respektive metod, men det är oklart om samma mönster håller för andra modeller. En större eller mer specialiserad modell kanske gynnar RAG mer, eller vice versa.

För att minska variation i modellens output sattes `temperature=0` i samtliga experiment, i linje med Microsofts riktlinjer för prompt engineering [13]. Varje konfiguration i parameterstudien kördes dessutom två gånger för att undersöka resultatets tillförlitlighet. Två körningar per konfiguration är däremot fortfarande ett begränsat underlag, vilket innebär att slumpmässig variation principiellt kan ha påverkat valet av den konfiguration som senare användes vid jämförelse mot RAG-pipelinen. De ytterligare körningar som senare genomfördes uppvisar däremot endast en variation på +/- 0.01 EM-enheter, vilket indikerar att modellens output var relativt stabil och att resultaten påverkades marginellt av slumpmässig variation. Jämförelsen mellan QA-systemet och RAG-pipelinen baserades på medelvärdet av tre oberoende körningar per system, vilket minskar risken för att tillfälliga variationer påverkat resultaten.

Few-shot-prompting användes inte i studien. Samtliga promptar utformades som zero-shot, det vill säga utan konkreta extraktionsexempel. Det går därmed inte att utesluta att båda systemen hade presterat bättre med fler välvalda exempel i prompten, och att den relativa skillnaden mellan metoderna hade sett annorlunda ut.

En vidare aspekt är att datasetet innehåller dokument på flera språk, däribland svenska, finska och spanska. Samtliga promptar skrevs på engelska, vilket kan ha påverkat extraktionskvaliteten för icke-engelska dokument om modellens förmåga att följa instruktioner varierar med dokumentets språk. Även om viss analys av språkets påverkan genomfördes var flera språk representerade av ett begränsat antal dokument, vilket gör det svårt att dra generella slutsatser om språkets faktiska påverkan på resultatet.

En ytterligare möjlig begränsning i utvärderingen är att endast fyra dokument i datasetet innehöll värden för *UltimateParent* och *DirectParent*. För övriga dokument motsvarade ett tomt svar det korrekta svaret. Detta innebär att modellerna i vissa fall kunde erhålla korrekt resultat genom att inte generera något svar alls, vilket kan ha bidragit till högre resultat för både det utvecklade QA-systemet och för RAG-pipelin. Eftersom samma utvärderingsmått och samma fält/frågor användes för båda system bedöms denna effekt påverka systemen på liknande sätt, vilket gör att jämförelsen mellan systemen fortfarande är relevant.

### 5.6 Hållbarhet, etik och samhällseliga aspekter

QA-systemet som utvecklats syftar till att automatisera extraktion av information från fondrelaterade dokument. En framgångsrik implementering skulle kunna reducera tid och kostnader kopplade till manuell datainsamling inom finanssektorn. Systemets utformning, där informationsutvinningen styrs genom naturligt språk, sänker den tekniska tröskeln för användning och anpassning, vilket möjliggör att domäner experter utan programmeringskunskap kan interagera med och förbättra systemet. En ökad automatiseringsgrad kan däremot påverka arbetsmarknaden negativt för personer som i dag arbetar med manuell datainsamling, och därmed vore det rimligt att utföra en kostnadsanalys i förhållande till manuell hantering för att bedöma systemets ekonomiska lönsamhet i produktionsmiljö.

Ur ett etiskt perspektiv bör det beaktas att språkmodeller är benägna att hitta fel information eller hallucinera genom att producera egna svar [2]. I ett finansiellt sammanhang kan felaktiga extraheringar få ekonomiska konsekvenser [4]. Systemet bör därför ses som ett stöd, och mänsklig granskning av utdata är rekommenderat. När det gäller miljöpåverkan visar forskning att större mängder bearbetad och genererad text är förknippade med ökad energiförbrukning vid LLM-inferens [6]. Det utvecklade QA-systemet skickar i sin bästa presterande konfiguration upp till 8000 tokens plus prompt per anrop, vilket innebär en större mängd indata jämfört med RAG-pipelin, som enbart skickar ett fåtal relevanta segment. Det utvecklade QA-systemet gör dessutom fler LLM-anrop per dokument: ett anrop per segment, ytterligare anrop under summeringssteget samt ett avslutande anrop vid frågebesvarandet. Utifrån tidigare forskning kring tokenmängd och inferenskostnad [6] innebär detta sannolikt en högre energiförbrukning jämfört med RAG-pipelin. Det noterades även empiriskt att varje körning tog längre tid i det utvecklade QA-systemet jämfört med vid RAG, vilket indirekt innebär en högre beräkningskostnad och därmed sannolikt även en större energiåtgång per dokument. En fullständig energijämförelse mellan de två systemen hade krävt mätning av faktisk energiförbrukning per dokument, vilket faller utanför ramen för denna studie.



## 6 Slutsatser

Arbetet undersökte en flerstegspipeline för frågebesvarande över finansiella dokument. Detta system optimerades först, och jämfördes sedan mot en välmotiverad konfiguration av ett traditionellt RAG-system.

Resultaten indikerar att det utvecklade QA-systemet generellt presterade bättre än den testade RAG-konfigurationen, både när det gäller Exact Match och F1-värde. Skillnaderna var störst för frågor relaterade till fund manager name och sub-funds. Resultaten indikerar att en pipeline bestående av frågefokuserad och hierarkisk sammanfattning av kontext kan vara fördelaktig vid frågebesvarande över finansiella dokument.

Parameterstudien visade att ökade segmentstorlekar förbättrade resultaten upp till 8000 tokens, medan ytterligare ökningsar till 12000 och 16000 tokens gav sämre resultat. Detta tyder på att större sammanhängande textsegment kan vara fördelaktiga upp till en viss nivå vid informationsutvinning från långa dokument.

Upprepade körningar av den bästa konfigurationen visade endast mindre variation mellan körningarna, vilket indikerar relativt stabil prestanda. Vid jämförelse med externa LLM-system presterade det utvecklade QA-systemet konkurrenskraftigt i relation till ChatGPT, Le Chat och Gemini på den aktuella dokumentuppsättningen.

Samtidigt innebär det utvecklade QA-systemet fler inferenssteg än ett traditionellt RAG-system, vilket kan påverka svarstid och resursförbrukning. Framtida arbete kan därför undersöka skillnader i exekveringstid och energiförbrukning mellan systemen, samt undersöka hur alternativa pipelinearkitekturer kan byggas.

Vidare kan framtida studier utvärdera varför prestandan försämrades vid segmentstorlekar större än 8000 tokens, samt analysera hur alternativa segmentstrategier påverkar resultaten.

Trots att den strukturerade och ostrukturerade PDF-läsaren gav snarlika resultat i denna studie, skulle framtida studier kunna undersöka om andra strukturerade PDF-läsare kan förbättra informationsextraktionen och därmed systemets övergripande prestanda.

Det vore även intressant att använda större och mer varierade dataset för att möjliggöra en mer robust analys av hur faktorer såsom dokumenttyp, dokumentlängd och språk påverkar systemens prestanda.

Även mänsklig utvärdering av domänexperter skulle kunna användas för att komplettera de automatiska måtten som använts för utvärdering av QA-systemet och RAG i denna studie, för att möjliggöra en mer nyanserad bedömning av modellernas predikterade svar.



## Källförteckning

1. Liu NF, Lin K, Hewitt J, Paranjape A, Bevilacqua M, Petroni F, m.fl. Lost in the Middle: How Language Models Use Long Contexts. *Trans Assoc Comput Linguist*. 23 februari 2024;12:157–73. doi:10.1162/tacl\_a\_00638
2. Barnett S, Kurniawan S, Thudumu S, Brannelly Z, Abdelrazek M. Seven Failure Points When Engineering a Retrieval Augmented Generation System. I: *Proceedings 2024 Ieee/Acm 3rd International Conference on Ai Engineering-Software Engineering for Ai, Cain 2024*. New York: Assoc Computing Machinery; 2024. s. 194–9. doi:10.1145/3644815.3644945
3. Lee J, Stevens N, Han SC, Song M. arXiv.org [Internet]. 2024 [citerad 11 maj 2026]. A Survey of Large Language Models in Finance (FinLLMs). Tillgänglig vid: <https://arxiv.org/abs/2402.02315v1> doi:10.1007/s00521-024-10495-6
4. Kang H, Liu XY. arXiv.org [Internet]. 2023 [citerad 11 maj 2026]. Deficiency of Large Language Models in Finance: An Empirical Examination of Hallucination. Tillgänglig vid: <https://arxiv.org/abs/2311.15548v1>
5. Wu H, Wang X, Fan Z. Addressing the sustainable AI trilemma: a case study on LLM agents and RAG [Internet]. arXiv; 2025 [citerad 08 april 2026]. Tillgänglig vid: <http://arxiv.org/abs/2501.08262> doi:10.48550/arXiv.2501.08262
6. Jegham N, Abdelatti M, Koh CY, Elmoubarki L, Hendawi A. How Hungry is AI? Benchmarking Energy, Water, and Carbon Footprint of LLM Inference [Internet]. arXiv; 2025 [citerad 11 maj 2026]. Tillgänglig vid: <http://arxiv.org/abs/2505.09598> doi:10.48550/arXiv.2505.09598
7. OpenAI Help Center [Internet]. [citerad 13 maj 2026]. What are tokens and how to count them? Tillgänglig vid: <https://help.openai.com/en/articles/4936856-what-are-tokens-and-how-to-count-them>
8. Prompt engineering | OpenAI API [Internet]. [citerad 06 maj 2026]. Tillgänglig vid: <https://developers.openai.com/api/docs/guides/prompt-engineering>
9. Khurana D, Koli A, Khatter K, Singh S. Natural Language Processing: State of The Art, Current Trends and Challenges. *Multimed Tools Appl*. januari 2023;82(3):3713–44. doi:10.1007/s11042-022-13428-4
10. Yao Y, Lin Z, Liu X, Li Y. Document Information Extraction in Engineering Reports Through Prompt Engineering with Large Language Models. I: *2025 IEEE 6th International Seminar on Artificial Intelligence, Networking and Information Technology (AINIT)* [Internet]. 2025 [citerad 31 mars 2026]. s. 1–4. Tillgänglig vid: <https://ieeexplore.ieee.org/document/11035034/> doi:10.1109/AINIT65432.2025.11035034
11. Sahoo P, Singh AK, Saha S, Jain V, Mondal S, Chadha A. arXiv.org [Internet]. 2024 [citerad 06 maj 2026]. A Systematic Survey of Prompt Engineering in Large Language Models: Techniques and Applications. Tillgänglig vid: <https://arxiv.org/abs/2402.07927v2>
12. OpenAI Help Center [Internet]. [citerad 06 maj 2026]. Best practices for prompt engineering with the OpenAI API. Tillgänglig vid: <https://help.openai.com/en/articles/6654000-best-practices-for-prompt-engineering-with-the-openai-api>

13. mrbullwinkle. Prompt engineering techniques - Microsoft Foundry [Internet]. [citerad 06 maj 2026]. Tillgänglig vid: <https://learn.microsoft.com/en-us/azure/foundry/openai/concepts/prompt-engineering>
14. Lewis P, Perez E, Piktus A, Petroni F, Karpukhin V, Goyal N, m.fl. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. I: Larochelle H, Ranzato M, Hadsell R, Balcan MF, Lin H, redaktörer. Advances in Neural Information Processing Systems 33, Neurips 2020. La Jolla: Neural Information Processing Systems (nips); 2020. (Advances in Neural Information Processing Systems).
15. Amazon Web Services, Inc. [Internet]. [citerad 21 april 2026]. What is RAG? - Retrieval-Augmented Generation AI Explained - AWS. Tillgänglig vid: <https://aws.amazon.com/what-is/retrieval-augmented-generation/>
16. Merola C, Singh J. Reconstructing Context Evaluating Advanced Chunking Strategies for Retrieval-Augmented Generation. I: Wang Z, Fang J, Frisoni G, Dai Z, Meng Z, Moro G, m.fl., redaktörer. Knowledge-Enhanced Information Retrieval, Keir 2025 [Internet]. Cham: Springer International Publishing Ag; 2026 [citerad 31 mars 2026]. s. 3–18. (Lecture Notes in Computer Science). Tillgänglig vid: [https://link.springer.com/chapter/10.1007/978-3-032-02899-0\\_1?utm\\_source=getftr&utm\\_medium=getftr&utm\\_campaign=getftr\\_pilot&getft\\_integrator=clarivate](https://link.springer.com/chapter/10.1007/978-3-032-02899-0_1?utm_source=getftr&utm_medium=getftr&utm_campaign=getftr_pilot&getft_integrator=clarivate) doi:10.1007/978-3-032-02899-0\_1
17. Gomez-Cabello CA, Prabha S, Haider SA, Genovese A, Collaco BG, Wood NG, m.fl. Comparative Evaluation of Advanced Chunking for Retrieval-Augmented Generation in Large Language Models for Clinical Decision Support. Bioeng-Basel. 01 november 2025;12(11):1194. doi:10.3390/biengineering12111194
18. Gutowska A. Chunking strategies for RAG tutorial using Granite | IBM [Internet]. 2025 [citerad 06 maj 2026]. Tillgänglig vid: <https://www.ibm.com/think/tutorials/chunking-strategies-for-rag-with-langchain-watsonx-ai>
19. Bennani S, Moslonka C. A Systematic Analysis of Chunking Strategies for Reliable Question Answering [Internet]. arXiv; 2026 [citerad 09 april 2026]. Tillgänglig vid: <http://arxiv.org/abs/2601.14123> doi:10.48550/arXiv.2601.14123
20. Mikolov T, Chen K, Corrado G, Dean J. Efficient Estimation of Word Representations in Vector Space [Internet]. arXiv; 2013 [citerad 31 mars 2026]. Tillgänglig vid: <http://arxiv.org/abs/1301.3781> doi:10.48550/arXiv.1301.3781
21. Reimers N, Gurevych I. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. I: 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (emnlp-Ijcnlp 2019): Proceedings of the Conference. Stroudsburg: Assoc Computational Linguistics-Acl; 2019. s. 3982–92.
22. Wang L, Yang N, Huang X, Yang L, Majumder R, Wei F. Multilingual E5 Text Embeddings: A Technical Report [Internet]. arXiv; 2024 [citerad 07 maj 2026]. Tillgänglig vid: <http://arxiv.org/abs/2402.05672> doi:10.48550/arXiv.2402.05672
23. Rethinking Chunk Size for Long-Document Retrieval: A Multi-Dataset Analysis [Internet]. [citerad 07 maj 2026]. Tillgänglig vid: <https://arxiv.org/html/2505.21700v2>
24. Best Practices for Common NVIDIA RAG Blueprint Settings — NVIDIA RAG blueprint [Internet]. [citerad 07 maj 2026]. Tillgänglig vid: [https://docs.nvidia.com/rag/2.4.0/accuracy\\_perf.html](https://docs.nvidia.com/rag/2.4.0/accuracy_perf.html)

25. haileytap. Chunk Documents - Azure AI Search [Internet]. [citerad 07 maj 2026]. Tillgänglig vid: <https://learn.microsoft.com/en-us/azure/search/vector-search-how-to-chunk-documents>
26. Radeva I, Popchev I, Dimitrova M. Similarity Thresholds in Retrieval-Augmented Generation. I: 2024 IEEE 12th International Conference on Intelligent Systems (IS) [Internet]. 2024 [citerad 07 maj 2026]. s. 1–7. Tillgänglig vid: <https://ieeexplore.ieee.org/document/10705214> doi:10.1109/IS61756.2024.10705214
27. Magesh V, Surani F, Dahl M, Suzgun M, Manning CD, Ho DE. arXiv.org [Internet]. 2024 [citerad 31 mars 2026]. Hallucination-Free? Assessing the Reliability of Leading AI Legal Research Tools. Tillgänglig vid: <https://arxiv.org/abs/2405.20362v1>
28. Li Z, Liu Y, Su Y, Collier N. Prompt Compression for Large Language Models: A Survey. I: Chiruzzo L, Wang L, redaktörer. Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies, Vol 1: Long Papers. Stroudsburg: Assoc Computational Linguistics-Acl; 2025. s. 7182–95.
29. Johnson W. The Compression Paradox in LLM Inference: Provider-Dependent Energy Effects of Prompt Compression [Internet]. arXiv; 2026 [citerad 17 april 2026]. Tillgänglig vid: <http://arxiv.org/abs/2603.23528> doi:10.48550/arXiv.2603.23528
30. Alanzi E, Alballaa S. Query-Focused Multi-document Summarization Survey. Int J Adv Comput Sci Appl. juni 2023;14(6):822–33.
31. Sinha N. QA-prompting: Improving Summarization with Large Language Models using Question-Answering [Internet]. arXiv; 2025 [citerad 30 mars 2026]. Tillgänglig vid: <https://arxiv.org/abs/2505.14347> doi:10.48550/ARXIV.2505.14347
32. Wu J, Ouyang L, Ziegler DM, Stiennon N, Lowe R, Leike J, m.fl. arXiv.org [Internet]. 2021 [citerad 30 mars 2026]. Recursively Summarizing Books with Human Feedback. Tillgänglig vid: <https://arxiv.org/abs/2109.10862v2>
33. Ou L, Lapata M. arXiv.org [Internet]. 2025 [citerad 30 mars 2026]. Context-Aware Hierarchical Merging for Long Document Summarization. Tillgänglig vid: <https://arxiv.org/abs/2502.00977v2>
34. What is a PDF? Portable Document Format | Adobe Acrobat [Internet]. [citerad 03 april 2026]. Tillgänglig vid: <https://www.adobe.com/acrobat/about-adobe-pdf.html>
35. Myasoedov VA, Vishnevskaya JA. Comparative Analysis of Text Extraction Methods from PDF Files for Subsequent Data Processing. I: 2026 ElCon Conference of Young Researchers in Electrical Engineering, Automation & Control Systems (ElCon-EE) [Internet]. 2026 [citerad 03 april 2026]. s. 118–21. Tillgänglig vid: <https://ieeexplore.ieee.org/document/11452828> doi:10.1109/ElCon-EE69794.2026.11452828
36. Lin D. arXiv.org [Internet]. 2024 [citerad 12 maj 2026]. Revolutionizing Retrieval-Augmented Generation with Enhanced PDF Structure Recognition. Tillgänglig vid: <https://arxiv.org/abs/2401.12599v1>
37. Docs by LangChain [Internet]. [citerad 07 maj 2026]. Document loader integrations. Tillgänglig vid: [https://docs.langchain.com/oss/python/integrations/document\\_loaders](https://docs.langchain.com/oss/python/integrations/document_loaders)
38. Raffel C, Shazeer N, Roberts A, Lee K, Narang S, Matena M, m.fl. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. J Mach Learn Res. 2020;21:140.

39. Rajpurkar P, Zhang J, Lopyrev K, Liang P. SQuAD: 100,000+ Questions for Machine Comprehension of Text [Internet]. arXiv; 2016 [citerad 06 april 2026]. Tillgänglig vid: <http://arxiv.org/abs/1606.05250> doi:10.48550/arXiv.1606.05250
40. Tam TYC, Sivarajkumar S, Kapoor S, Stolyar AV, Polanska K, McCarthy KR, m.fl. A framework for human evaluation of large language models in healthcare derived from literature review. *Npj Digit Med*. 28 september 2024;7(1):258. doi:10.1038/s41746-024-01258-7
41. Gana B, Allende-Cid H, Rüping S, Becerra-Rozas M, Zamora J. A systematic review of long document summarization methods: Evaluation metrics and approaches. *Neurocomputing*. 28 november 2025;655:131287. doi:10.1016/j.neucom.2025.131287



