THOMAS MORE

STAGE WISEMEN

Project Plan

Seppe Stroobants

2023 - 2024

Inhoud

1. Introductie RAG

1.1. Korte introductie van stage-opdracht

Deze stage richt zich op een onderzoek naar Retrieval-Augmented Generation (RAG) en de toepassing ervan binnen kunstmatige intelligentie. Het doel is een **proof-of-technology (POT)** te ontwikkelen die aantoont hoe RAG de kwaliteit en relevantie van AI-gegenereerde content kan verbeteren door retrieval- en generatieve modellen te combineren.

De focus ligt op **backend-ontwikkeling**, waarbij verschillende RAG-modellen, methodologieën en tools worden geëvalueerd. Voor prototyping wordt **Python** gebruikt, terwijl de uiteindelijke implementatie plaatsvindt in een **Node.js/Nest.js backend** met **TypeScript** in combinatie met **Python** voor functies die een Python werkomgeving nodig hebben, **PostgreSQL** met **PGVector** faciliteren de data-opslag en retrieval, terwijl **Docker** wordt ingezet voor DevOps en deployment.

De kernvraag van dit onderzoek luidt: **"Hoe kan Retrieval-Augmented Generation (RAG) worden toegepast om de kwaliteit en efficiëntie van AI-gegenereerde content te verbeteren?"** Dit wordt onderzocht aan de hand van een webtoepassing die RAG in de praktijk demonstreert, waarbij verschillende implementaties worden geëvalueerd en gedocumenteerd in een technische analyse.

1.2. Doel van plan van aanpak

Dit document biedt een **duidelijke structuur** voor de uitvoering van de stageopdracht. Het beschrijft de doelstellingen, methodologie, planning en benodigde middelen. Daarnaast fungeert het als een **communicatiemiddel** tussen de stagiair, de begeleiders en het stagebedrijf.

1.3. Wie zijn betrokken?

De stage wordt uitgevoerd bij **Wisemen**, een Belgisch digitaal agency gespecialiseerd in strategie, ontwikkeling en marketing. Sinds de oprichting in 2013 (voorheen bekend als Appwise) is Wisemen uitgegroeid tot een full-service digital agency.

Diensten en Expertise:

- **Digitale Transformatie**: Wisemen ontwikkelt strategieën die bedrijfsdoelstellingen afstemmen op de behoeften van klanten en gebruikers, met oog voor technische mogelijkheden.
- **Design**: Het bureau ontwerpt gebruiksvriendelijke ervaringen en interfaces volgens een eigen design-driven aanpak, waardoor producten en diensten aantrekkelijk en effectief zijn.
- **Softwareontwikkeling**: Wisemen biedt maatwerksoftware, innovatieve websites en marketingstrategieën die efficiëntie bevorderen, klanten betrekken en groei stimuleren.
- **Marketing**: Met geïntegreerde marketingcampagnes helpt Wisemen merken om zich te onderscheiden en hun doelgroep effectief te bereiken.

Begeleiders en teamleden:

- **Yano Stulens** – Functioneel Analist & Stagebegeleider
- **Gerrit Schalenbourg** – Head of Operations
- **Maarten Sijmkens** – Data Engineer
- **Kristine Mangelschots** – Docent & Stagebegeleider (Thomas More)
- **Seppe Stroobants** – Stagiair Bachelor Toegepaste Informatica

2. Doelstelling RAG

2.1. Wat willen we bereiken met deze opdracht

Het hoofddoel is om een **proof-of-technology** te ontwikkelen die aantoont hoe RAG kan bijdragen aan **betere AI-systemen** door ze toegang te geven tot actuele, interne bedrijfsinformatie.
Concreet streven we naar:

- Integratie van RAG binnen een bestaande IT-infrastructuur.
- Ontwikkeling van een functionele demo voor interne kennisdeling en klantenservice.
- Analyse van de meest geschikte technologieën en methodologieën binnen Wisemen

2.2. Verwachte eindresultaat

Een werkend **RAG-systeem**, dat:

- Bedrijfsinformatie **ophaalt, verwerkt en weergeeft** met bronvermelding.
- **Toegangscontrole en privacy** respecteert.
- Specifieke use cases ondersteunt, zoals interne kennisbanken.
- Code bevat die **leesbaar, onderhoudbaar en getest** is.

3. Opdrachtomschrijving RAG

3.1. Wat houdt de opdracht precies in?

De stage richt zich op het ontwikkelen van een **proof-of-technology** voor **RAG in een bedrijfsomgeving**. De webapplicatie zal laten zien hoe AI bedrijfsinformatie kan benutten voor nauwkeurigere en relevantere gegenereerde content.

De focus ligt op:

- **Backend-ontwikkeling** met **Node.js/Nest.js** en **TypeScript**.
- **Dataverwerking** met **PostgreSQL** met **PGVector** voor efficiënte opslag en retrieval.
- **Privacy** en **toegangscontrole** zodat gevoelige bedrijfsinformatie niet extern toegankelijk is.
- Technische analyse van verschillende **RAG-modellen** en **optimalisatiemethoden**.

3.2. Welke problemen lost het op?

Veel bedrijven willen AI inzetten, maar lopen tegen drie grote uitdagingen aan:

1. **Verouderde kennis** – AI-modellen zoals ChatGPT gebruiken statische trainingsdata en missen recente of bedrijfsspecifieke informatie.
2. **Gevoelige data** – Standaard AI-oplossingen slaan gegevens extern op, wat privacy- en compliance-risico's met zich meebrengt.
3. **Toegangscontrole** – AI moet rekening houden met wie welke informatie mag inzien binnen een bedrijf.

Met **RAG** kunnen deze problemen worden aangepakt door AI-modellen te laten zoeken in up-to-date, interne informatiebronnen, zonder bedrijfsgevoelige data bloot te stellen aan externe systemen.

3.3. Verwachtingen van het stagebedrijf

Wisemen verwacht dat deze stageopdracht leidt tot:

- Een **werkende proof-of-technology** die RAG toepast op realistische bedrijfsdata.
- **Duidelijke inzichten** in de haalbaarheid en voordelen van RAG binnen een zakelijke context.
- Een **technische analyse** met aanbevelingen voor implementatie bij klanten van Wisemen.
- Een **documentatie en evaluatie** van de gebruikte technologieën, methoden en uitdagingen.

Daarnaast biedt deze opdracht **nieuwe businesskansen** voor Wisemen, door AI-oplossingen op maat te ontwikkelen voor bedrijven die worstelen met interne kennisdeling, klantenservice en compliance.

4. Aanpak en werkwijze RAG

4.1 Welke stappen neem je om de opdracht uit te voeren?

De ontwikkeling verloopt in verschillende fasen:

1. **Onderzoek & Oriëntatie**
    - Verkennen van RAG-technologie en bestaande implementaties.
    - Analyseren van bestaande LLM-modellen en retrieval-methoden.
    - Bepalen van de juiste infrastructuur en tools.
2. **Technische Setup & Prototyping**
    - Opzetten van een ontwikkelomgeving met **Node.js/Nest.js** en **TypeScript**.
    - Implementeren van **text embeddings** met Hugging Face om tekstuele data om te zetten naar vectoren.
    - Integreren van **Ollama** en het **Llama 3.2-model** als LLM binnen de RAG-pipeline.
    - Configureren van **PostgreSQL** en **vector databases** voor efficiënte opslag en retrieval.
3. **Proof-of-Concept (PoC) Ontwikkeling**
    - Bouwen van een backend die interne bedrijfsinformatie kan ophalen en verwerken.
    - Implementeren van een **RAG-pipeline** die queries interpreteert, relevante data zoekt en combineert met LLM-antwoorden.
    - Testen en optimaliseren van de pipeline op basis van snelheid, nauwkeurigheid en schaalbaarheid.
4. **Validatie & Evaluatie**
    - Uitvoeren van tests met realistische bedrijfsdata.
    - Benchmarking van de prestaties en betrouwbaarheid van het systeem.
    - Documenteren van de bevindingen en aanbevelingen voor verdere optimalisatie.
5. **Oplevering & Presentatie**
    - Afronden van de proof-of-technology en documentatie.
    - Presenteren van de resultaten aan Wisemen en mogelijke stakeholders.
    - Bespreken van verdere toepassingen en implementatiemogelijkheden.

4.1. Welke methodes/tools gebruik je?

Voor de implementatie van deze opdracht worden de volgende methodes en tools gebruikt:

- **LLM & AI-modellen**:
  - **Ollama** als uitvoeringsomgeving.
  - **Llama 3.2** als het taalmodel in de RAG-pipeline.
  - **Hugging Face text embeddings** voor het omzetten van tekstuele data naar vectoren.
- **Backend & Dataverwerking**:
  - **Node.js/Nest.js** voor backend-ontwikkeling.
  - **PostgreSQL + PGVector** voor opslag en retrieval.
  - **Python:** voor AI gerelateerde functies toe te passen
- **DevOps & Deployment**:
  - **Docker** voor containerisatie en schaalbaarheid.

5. Onderzoeksfase RAG

Tijdens de onderzoeksfase heb ik me gericht op het testen en selecteren van geschikte **Large Language Models (LLM's)**, het bepalen van een efficiënte **vector database** en het ontwikkelen van **API-endpoints** voor dataopslag en retrieval. Dit onderzoek vormt de technische basis voor de implementatie van de **Retrieval-Augmented Generation (RAG)**-pipeline.

5.1. Testen en selecteren van Large Language Models (LLM's)

Om de prestaties en geschiktheid van verschillende LLM's te evalueren, heb ik tests uitgevoerd met **Ollama** als uitvoeringsomgeving. De volgende modellen werden getest:

- **llama3.2-vision:latest**
- **qwq:32b**
- **deepseek-r1:latest**
- **llama3.2:latest**

5.1.1. Evaluatiecriteria

Bij de beoordeling van de modellen heb ik gelet op de volgende aspecten:

- **Nauwkeurigheid:** Hoe goed begrijpt en genereert het model relevante antwoorden op basis van retrieval-data?
- **Snelheid:** Hoe snel verwerkt het model zoekopdrachten en genereert het resultaten?
- **Geheugengebruik:** Hoeveel systeembronnen vereist het model om soepel te draaien?
- **Efficiëntie bij korte queries:** Hoe goed presteert het model bij eenvoudige en directe zoekopdrachten?

Resultaten
Uit de tests bleek dat **llama3.2** de beste balans bood tussen nauwkeurigheid, snelheid en efficiëntie. Het model presteerde uitstekend bij korte queries en had een relatief lage systeembelasting vergeleken met **qwq:32b**, dat weliswaar goed presteerde, maar met een **modelgrootte van 19GB** te zwaar was voor optimale integratie. Om deze reden heb ik gekozen voor **llama3.2** als het primaire model voor tekstuele data.

5.2. Onderzoek naar Vector Databases

Om retrieval efficiënt en schaalbaar te maken, is het essentieel om tekstuele data als **vector embeddings** op te slaan in een gespecialiseerde database. Hiervoor heb ik twee populaire vector databases onderzocht:

- **PGVector** – een uitbreiding op PostgreSQL, waardoor het goed integreert met bestaande systemen.
- **ChromaDB** – een dedicated open-source vector database die speciaal is ontworpen voor high-performance retrieval.

5.2.1. Evaluatie en keuze

De belangrijkste evaluatiecriteria waren:

- **Compatibiliteit:** Hoe goed past de database binnen de bestaande **tech stack van Wisemen**?
- **Prestaties:** Hoe efficiënt worden vectoren opgeslagen en opgehaald?
- **Schaalbaarheid:** Hoe eenvoudig is de database uit te breiden bij grotere datasets?

Na een korte **setup-test** en intern overleg is gekozen voor **PGVector**, omdat:

- **Wisemen al met PostgreSQL werkt**, waardoor integratie eenvoudig is.
- **PGVector een native uitbreiding van PostgreSQL is**, wat betekent dat er geen extra infrastructuur nodig is.
- **Vector storage in PGVector eenvoudig te implementeren is** via een extra tabel, zonder de bestaande database-architectuur te verstoren.

5.3. Ontwikkeling van API-endpoints voor Data Embedding en Retrieval

Om interactie met de vector database mogelijk te maken, heb ik vier API-endpoints ontwikkeld in **Python**. Deze endpoints zorgen voor de integratie van de RAG-pipeline met PGVector.

5.3.1. Overzicht van API-endpoints

1. **Embedding van data** in PGVector: tekst wordt omgezet in vectoren en opgeslagen in de database.
2. **Verwijderen van documenten** uit de vector database.
3. **Opvragen van een lijst met opgeslagen documenten** om inzicht te krijgen in beschikbare data.
4. **Uitvoeren van een search-query** op basis van vector overeenkomsten.

Deze API's vormen de basis voor **informatie-opslag en retrieval binnen de RAG-pipeline**, zodat relevante informatie snel kan worden opgehaald en gecombineerd met AI-gegenereerde antwoorden.

5.4. Onderzoek naar Embedding Modellen

Om tekstuele data om te zetten naar hoogwaardige **vector embeddings**, heb ik verschillende **Sentence Transformer-modellen** uit de **Hugging Face-library** getest. De twee meest veelbelovende modellen waren:

- **all-MiniLM-L6-v2**
- **all-mpnet-base-v2**

5.4.1. Test en evaluatie van embeddings

Ik heb een test geschreven om de prestaties van verschillende embeddings te vergelijken op basis van:

- **Retrieval-prestaties:** Hoe nauwkeurig en snel worden relevante documenten opgehaald?
- **Efficiëntie:** Hoeveel resources worden verbruikt bij het genereren van embeddings?
- **Schaalbaarheid:** Hoe presteert het model bij grotere hoeveelheden tekstdata?

Uit de test bleek dat **all-MiniLM-L6-v2** een **50% snellere retrieval-performance** bood in vergelijking met **all-mpnet-base-v2**. Hierdoor is **all-MiniLM-L6-v2** gekozen als het embedding-model voor de implementatie.

5.5. Conclusie van de Onderzoeksfase

De onderzoeksfase heeft geleid tot de volgende technische keuzes:

- **LLM: llama3.2** als de meest efficiënte optie voor tekstuele query's.
- **Vector database: PGVector**, vanwege de eenvoudige integratie met PostgreSQL.
- **API-endpoints:** Ontwikkeld in **Python** voor embedding, documentbeheer en retrieval.
- **Embedding model: all-MiniLM-L6-v2**, vanwege de hoge retrieval-efficiëntie en prestaties.

Deze keuzes vormen de fundering voor de verdere ontwikkeling van de **Proof-of-Technology (PoT)** en zorgen voor een schaalbare en efficiënte implementatie van de RAG-pipeline.

6. Introductie E-mailclassificatie

6.1. Korte introductie van stage-opdracht

Dit deel van de stage richt zich op de ontwikkeling van een pipeline die automatisch inkomende e-mails classificeert, analyseert en opvolgt met behulp van Large Language Models (LLM's). Het project onderzoekt hoe generatieve AI ingezet kan worden om repetitieve taken, zoals het lezen, interpreteren en beantwoorden van e-mails, te automatiseren binnen een zakelijke context.

De toepassing is opgebouwd rond drie AI-modellen (OpenAI GPT, Google Gemini, en Ollama Gemma) en ondersteunt zowel Gmail als Outlook als e-mailprovider. Classificaties worden uitgevoerd in gestructureerde JSON-output met kerninformatie, zoals categorie, aanbevolen acties en orderdetails.

De pipeline is uitbreidbaar naar ERP-integratie, waarbij geclassificeerde e-mails automatisch acties kunnen triggeren binnen bedrijfsprocessen zoals orderverwerking of supportticketing.

De centrale onderzoeksvraag luidt:
"Hoe kan een AI-gedreven pipeline voor e-mailclassificatie en opvolging de efficiëntie verhogen in de verwerking van inkomende e-mailcommunicatie?"

6.1.1. Doel van plan van aanpak

Dit plan van aanpak biedt een helder overzicht van de werkwijze, fasering en technische keuzes voor het e-mailclassificatieproject. Het dient als communicatiedocument tussen de stagiair, Wisemen en de betrokken opleidingsinstantie, en waarborgt een gestructureerde en haalbare uitvoering van het project.

6.1.2. Wie zijn betrokken?

De opdracht wordt uitgevoerd binnen het team van Wisemen voor een van hun klanten.

Begeleiders en teamleden:

- Yano Stulens – Functioneel Analist & Stagebegeleider
- Gerrit Schalenbourg – Head of Operations
- Maxime Vanheusden
- Kristine Mangelschots – Docent & Stagebegeleider (Thomas More)
- Seppe Stroobants – Stagiair Bachelor Toegepaste Informatica

6.2. Doelstelling e-mailclassificatie

6.2.1. Wat willen we bereiken met deze opdracht

Het hoofddoel is het realiseren van een werkende AI-pipeline die automatisch e-mails classificeert en opvolgt.
Concreet streven we naar:

- Inzicht in de prestaties van generieke LLM's bij natuurlijke taalclassificatie.
- Extractie van contextuele gegevens uit e-mails in een gestructureerd formaat.
- Automatische generatie van conceptantwoorden en ontvangstbevestigingen.
- Integratie met zowel Gmail als Outlook via hun officiële API's.
- Voorbereiding op toekomstige integratie met ERP- en CRM-systemen.

6.2.2. Verwachte eindresultaat

Een volledig werkende prototypepipeline, die:

- Ongelezen e-mails automatisch classificeert met behulp van meerdere LLM-methodes.
- Output levert in gestructureerde JSON (categorie, confidence, actie, etc.).
- Draft-antwoorden genereert op basis van de context.
- Bevestigingen automatisch verstuurt op basis van classificatie.
- API-koppelingen implementeert met Gmail en Outlook.
- Documentatie bevat over logica, opbouw, prompts, validatie en beveiliging.

6.3. Opdrachtomschrijving e-mailclassificatie

6.3.1. Wat houdt de opdracht precies in?

De stageopdracht bestaat uit de ontwikkeling van een e-mailverwerkingspipeline. De pipeline leest ongelezen e-mails uit gekoppelde inboxen (Gmail of Outlook), classificeert de inhoud, en voert automatisch acties uit zoals het opstellen van antwoorden en het verzenden van bevestigingen.

De verwerking bestaat uit:

- Gebruik van generatieve AI om e-mails te interpreteren.
- Structureren van output in JSON via Pydantic.
- Draft-opbouw en verzending via API's van Gmail/Outlook.
- Logging en validatie van resultaten voor foutafhandeling en benchmarking.

6.3.2. Welke problemen lost het op?

Bedrijven krijgen dagelijks honderden tot duizenden e-mails.
De volgende problemen worden aangepakt:

1. **Tijdrovende verwerking** – Menselijke verwerking is traag en foutgevoelig.
2. **Inconsistente classificatie** – Verschillende medewerkers kunnen mails anders interpreteren.
3. **Trage opvolging** – Klanten ontvangen geen directe bevestiging of reactie.
4. **Slechte schaalbaarheid** – Volume neemt toe, zonder lineaire personeelsgroei.

AI-gebaseerde classificatie brengt automatisering, snelheid en consistentie in dit proces.

6.3.3. Verwachtingen van het stagebedrijf

Wisemen verwacht dat deze opdracht leidt tot:

- Een functioneel prototype van een e-mailclassificatiepipeline.
- Inzicht in de prestaties van cloud-based en lokale LLM's.
- Evaluatie van API-koppelingen met externe systemen (e-maildiensten).
- Concrete uitbreidbare codebase die kan dienen als basis voor klantprojecten.

6.4. Aanpak en werkwijze e-mailclassificatie

6.4.1. Welke stappen neem je om de opdracht uit te voeren?

Het project is uitgevoerd in vijf iteratieve fases:

1. **Onderzoek & Verkenning**
    - Analyse van Gmail & Outlook API's
    - Test met LLM's (Gemini, OpenAI, Ollama)
    - Structurering van gewenste JSON-output
2. **Technische Setup**
    - Implementatie OAuth-flow voor Gmail/Outlook
    - Opbouw Python-classificatieservice met JSON-validatie
    - Ontwerp van prompts voor elk model
3. **Pipeline Implementatie**
    - Classificatie + extractie van key info
    - Automatische opbouw van conceptantwoorden
    - Ontvangstbevestigingen afhankelijk van categorie
4. **Benchmarking & Validatie**
    - Testen van modeloutput op betrouwbaarheid, snelheid en correctheid
    - Validatie van JSON met Pydantic
    - Logging en foutafhandeling
5. **Resultaat & Voorstel tot integratie**
    - Documentatie & presentatie van pipeline
    - Toekomstvoorstel: ERP-integratie & UI-dashboard

6.4.2. Welke methodes/tools gebruik je?

LLM & AI-modellen:

- Google Gemini (gemini-1.5-flash-001 via Generative AI API)

- OpenAI GPT-4.1-nano (via openai Python SDK)
- Ollama met lokaal Gemma 4B model

Validatie & Structurering:

- Pydantic voor validatie van JSON-output
- Regex-cleaning bij parsing foutieve modeloutput

Back-end & Automatisering:

- Python + FastAPI
- Gmail API + google-auth
- Outlook API via Microsoft Graph + msal
- Draft generatie en verzending via API-calls
- Logging, fallback-strategie, error recovery