**Waarom Retrieval-Augmented Generation de beste keuze is**

**Inleiding**

Dit document legt uit waarom Retrieval-Augmented Generation (RAG) de meest geschikte oplossing is voor onze LLM. In dit document bespreken we hoe RAG werkt, de voordelen ten opzichte van andere methoden zoals fine-tuning en prompt engineering, en waarom het goed aansluit voor onze LLM.

**Wat is Retrieval-Augmented Generation (RAG)?**

Retrieval-Augmented Generation (RAG) is een methode die een getraind taalmodel, In dit document zullen we een model van ChatGPT als voorbeeld gebruiken, combineert met een mechanisme. Dit mechanisme haalt relevante data op uit externe databronnen, zoals databases, documenten of API’s, en voegt deze toe als context voor het taalmodel. Hierdoor hoeft het taalmodel niet alle mogelijke informatie te leren, in plaats daarvan verwerkt het dynamisch relevante informatie om vragen te beantwoorden of rapportages te genereren.

**Hoe verschilt RAG van andere methoden?**

* Geen vooraf ingestelde training nodig: In tegenstelling tot fine-tuning leert het model geen nieuwe data, maar gebruikt het real-time relevante informatie.
* Houdt data actueel: Doordat RAG gebruikmaakt van externe bronnen, werkt het altijd met de nieuwste data die is ingeladen, eventueel kan het ook met gecachete data werken.
* Grote datasets: RAG kan omgaan met zeer grote datasets door alleen relevante stukken data naar het model te sturen.

**Hoe werkt RAG?**

RAG werkt in vier belangrijke stappen:

1. Embedding genereren

Een embedding is een numerieke representatie die het model helpt om de betekenis en context van een stuk data te begrijpen. Voor elk stuk data in je dataset genereer je een embedding met een embedding-model, zoals bijvoorbeeld het text-embedding-ada-002 van OpenAI.

1. Opslag in een vector database

De gegenereerde embeddings worden opgeslagen in een vector database, een gespecialiseerd type database dat efficiënt kan zoeken naar de meest relevante stukken data. Veelgebruikte vector databases zijn:

* Pinecone
* Weaviate
* FAISS (open-source en lokaal te gebruiken)

1. Zoekproces

Wanneer een gebruiker een vraag stelt, wordt van die vraag ook een embedding gegenereerd. De vector database zoekt vervolgens naar stukken data die het meest overeenkomen met de vraag-embedding.

1. Context toevoegen aan het model

De relevante stukken data worden gecombineerd tot een context. Deze context wordt samen met de vraag van de gebruiker naar het model word gestuurd, waardoor het model een specifiek, contextueel antwoord kan genereren.

**Waarom is RAG de beste keuze?**

1. Flexibiliteit

RAG kan dynamisch inspelen op veranderingen in de data. Omdat de data in een externe vector database staat, kan je eenvoudig nieuwe databronnen toevoegen of bestaande data updaten zonder het model opnieuw te trainen. Dit maakt RAG veel flexibeler dan fine-tuning, waarbij alle nieuwe data expliciet in het model moet worden opgenomen.

1. Gebruik van actuele data

Bij fine-tuning wordt het model getraind op een specifieke dataset, wat betekent dat de kennis daarbuiten gaat. Dit is een nadeel in situaties waarin data regelmatig verandert. Met RAG blijft de LLM altijd actueel, omdat het model data ophaalt uit externe bronnen die voortdurend kunnen worden bijgewerkt.

1. Grote hoeveelheden data

RAG kan efficiënt omgaan met grote hoeveelheden data. Het model hoeft niet alles tegelijk te verwerken; alleen de meest relevante informatie wordt opgehaald en verwerkt. Dit maakt het mogelijk om grote datasets, zoals uitgebreide logs of gedetailleerde rapportages, te gebruiken zonder de performance van het model te beïnvloeden.

**Vergelijking met andere methoden**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Methode | Flexibiliteit | Kosten | Gebruik van actuele data | Complexiteit |
| Fine-tuning | Laag | Hoog | Nee | Hoog |
| Prompt engineering | Hoog | Laag | Beperkt | Laag |
| RAG | Hoog | Gemiddeld | Ja | Gemiddeld |

**Waarom past RAG in deze use case?**

Bij deze use case is de LLM bedoeld om:

1. Meerdere databronnen te analyseren: Bijvoorbeeld Nederlandse en Europese bestanden
2. Flexibele rapportages te genereren: Gebruikers kunnen vragen stellen over verschillende onderwerpen.
3. Werken met dynamische datasets: Data verandert regelmatig, wat vraagt om een systeem dat makkelijk kan worden bijgewerkt.

RAG past daar perfect, omdat:

* Het makkelijk nieuwe databronnen kan integreren.
* Het geen dure of tijdrovende fine-tuning vereist.
* Het dynamisch relevante data ophaalt, waardoor gebruikers altijd de meest actuele data terug krijgen.
* Het goed om kan gaan met grote datasets, zoals gedetailleerde logbestanden.

**Conclusie**

Retrieval-Augmented Generation is een efficiënte en flexibele aanpak voor het bouwen van LLM die de juiste data teruggeeft met betrekking tot de energie transitie op basis van meerdere databronnen. Het maakt optimaal gebruik van de kracht van een LLM zonder de nadelen van fine-tuning of beperkingen met prompt engineering. RAG biedt de perfecte balans tussen prestaties en toekomstbestendigheid, waardoor het de beste keuze is voor deze use case.

**Bronnenlijst**

* Carnes, B. (2024, 15 mei). *Learn RAG from Scratch – Python AI Tutorial from a LangChain Engineer*. freeCodeCamp.org. <https://www.freecodecamp.org/news/mastering-rag-from-scratch#:~:text=Retrieval-Augmented%20Generation%20%28RAG%29%20is%20a%20powerful%20framework,that%20integrates%20retrieval%20into%20the%20sequence%20generation%20process>
* Monigatti, L. (2024, 24 juni). Retrieval-Augmented Generation (RAG): From Theory to LangChain Implementation. *Medium*. <https://towardsdatascience.com/retrieval-augmented-generation-rag-from-theory-to-langchain-implementation-4e9bd5f6a4f2>
* Yadav, A. (2024, 19 november). Prompt Engineering vs. RAG vs. Finetuning: What’s the Difference? *Medium*. <https://medium.com/aidetic/prompt-engineering-vs-rag-vs-finetuning-whats-the-difference-78466d70b5df#:~:text=Prompt%20engineering%20offers%20a%20cost-effective%20and%20flexible%20approach,responses%20by%20integrating%20retrieval%20mechanisms%20with%20generative%20models>
* Pandey, V. (2023, 19 augustus). Should you Prompt, RAG, Tune, or Train? A Guide to Choose the Right Generative AI Approach. *Medium*. <https://medium.com/@pandey.vikesh/should-you-prompt-rag-tune-or-train-a-guide-to-choose-the-right-generative-ai-approach-5e264043bd7d>