AI Assistenten – maatregelen voor een juist antwoord

Bijna iedere AI Assistent is anders, want er zijn veel variabelen:

- De data is anders: (on)gestructureerd, verschillende bronnen, verschillende kwaliteit
- Het doel is anders
- De behoefte is anders

Dit zorgt ervoor dat er geen 1 oplossing is om een AI Assistent juiste antwoorden te laten geven.

Om over na te denken (geen oordeel):

Als iemand de Belastingdienst belt met een vraag, wordt het antwoord door de medewerker dan gecontroleerd door iemand anders?

Het begint met je data

- Data kan eenvoudig zijn om mee te werken: korte documenten met een enkel onderwerp erop, makkelijk te destilleren.
- Moeilijker om mee te werken: lange documenten met complexe onderwerpen, vaktermen, etc.
- Relationele data, die aan elkaar gekoppeld moet worden
- ....

Je data bepaalt of je een Knowledge Graph (relationele database) moet gebruiken of een gewone database.

Data bewerken

- Data opschonen: Voer datavalidatie en -opschoning uit om dubbelingen, verouderde informatie en tegenstrijdigheden te verwijderen. Dit voorkomt dat onjuiste of inconsistent data de antwoorden beïnvloeden.
    - Met HTML: sluit ruis (menu-items, footer, etc) uit en selecteer alleen de content
- Structurering: Organiseer data in een gestructureerd formaat met metadata (bijv. onderwerp, auteur, documenttype) en hiërarchieën om de opzoekbaarheid en contextuele relevantie te verbeteren.
- Normalisatie: Zorg voor consistente dataformaten en representaties om naadloze integratie en verwerking te garanderen.
- Identificeer metadata die kunnen helpen met filteren of verrijken van je chunks. Zo kan je bijvoorbeeld iedere chunk van dezelfde pagina verrijken met de H1/paginatitel om de semantische relevantie te verhogen.

Data chunking

RAG werkt met data die niet te groot is (chunks), dus geen pagina van 10 A4tjes, maar misschien wel 10 stukken voor die ene pagina.
Chunking kan op vele manieren, zoals semantisch, agentic, recursieve en vaste grootte. De manier die je kiest hangt af van je data en moet je ook testen voor jouw data (en mee experimenteren).

Data voor generatie

Belangrijk:
Zodra je informatie extraheert uit een document en omzet naar gewoon tekst, dan worden bv. tabellen onleesbaar en ben je de (hiërarchische) informatie van bv. headings (H1, H2, H3,..) en opsommingen kwijt.
Zorg er dus voor dat je de informatie uit het originele document omzet naar Markdown. Zo wordt deze belangrijke informatie bewaard en overgedragen aan de LLM.

Embedding

Normaal gesproken combineer je keyword-zoeken met semantisch zoeken (embeddings).
Kies een embedding model die goed werkt voor jouw data. Houd rekening met de taal (sommige embeddingsmodellen zijn vooral Engels gefocust). En kies een model die zowel woord als zins-embedding ondersteunt.

Chunk en embedding evaluatie

Evalueer chunk-kwaliteit om de optimale chunk-grootte en strategie te bepalen.

Evalueer embedding-kwaliteit om te controleren dat je embedding-model de juiste is.

De input van de gebruiker

De input van de gebruiker moet samen met de chatgeschiedenis geanalyseerd worden om de intentie van de huidige input te achterhalen.

Deze intentie bepaalt wat je volgende stap is:

- Direct antwoorden, bv. bij “hallo”, “dank je wel”, maar ook als de intentie off-topic is (“hoe maak ik uiensoep” als vraag voor Rijksoverheid), of andere situaties waarbij je geen documenten nodig hebt.
- Antwoord opzoeken

Antwoord zoeken

De intentie kan geoptimaliseerd worden voor zoeken in jouw database, bijvoorbeeld door toevoegen van bepaalde keywords, filters toepassen, etc. Afhankelijk van je data kan soms helpen om te zoeken op meerdere varianten van dezelfde vraag (voor keyword matching). En soms moet je een gecombineerde vraag (bv. “Hoeveel is de AOW en wanneer wordt het uitbetaald”) splitsen in meerdere vragen.
Hierbij kan het ook handig zijn om extra context toe te voegen die de LLM helpt de beste zoekvraag of zoekvragen te formulieren.

Gebruik voorbeelden

Gebruik voorbeelden van query’s die de LLM helpen om een goede query te formuleren of juist een foute query te herkennen.

Gewicht

Afhankelijk van je data en de doelgroep kan je ook bijvoorbeeld meer gewicht leggen op semantisch zoeken (meer algemene vragen, minder specialistisch/vakinhoudelijk) of juist op keyword zoeken (bv. wanneer de gebruiker exact weet wat deze zoekt en er bv. veel vaktermen gebruikt worden).

Reranker, fusion, ...

De resultaten van zowel het vector-zoeken en keyword-zoeken moeten samen komen. Dat kan bv. met RRF (Reciprocal Rank Fusion), maar ook met een Reranker. Een Reranker is een LLM die getraind is om inhoud van documenten te scoren op de query naar relevantie. Let hier ook weer op de taal die ondersteund wordt.

Selectie

Zet een minimum score (op basis van je testen) voor zowel vector-zoeken en keyword-zoeken. Je wilt niet altijd de top 15 (of hoeveel chunks/documenten je kiest per zoek-methode), zodat documenten die te laag scoren niet mee gaan en er minder ruis is.

Ditzelfde kun je daarna ook doen voor de RRF of de Reranker (of andere methode van samenvoegen). Probeer ervoor te zorgen dat de LLM die de output moet genereren zoveel mogelijk alleen relevante informatie ziet en ook niet teveel (hang ook van de context-window af en hoe goed de LLM is).

Evaluatie

Gebruik bestaande evaluatiemiddelen of maak je eigen. Maak bijvoorbeeld een FAQ van 20 vragen per domein, stel van tevoren vast welke chunks **altijd** gevonden moeten worden per vraag en automatiseer een check dat die chunks ook met iedere iteratie over de FAQ gevonden wordt.

Antwoordgeneratie

Instrueer de chatbot dat deze alleen antwoord geeft als het antwoord in de bronnen staat en geen eigen kennis gebruikt. En geef een escape dat deze anders antwoordt met iets als "Ik heb geen antwoord kunnen vinden.."

Geef goede instructies voor tone-of-voice, taalniveau, empathie, etc.

Gebruik voorbeelden

Gebruik voorbeelden van responses die de LLM helpen om een goede response te formuleren of juist een foute response te herkennen.

Automatische evaluatie

Het is mogelijk om een evaluatie toe te voegen die de output evalueert op bv. “volledigheid”, tone-of-voice”, “confidence score”, “juiste info”, etc.

Hier kan je thresholds op zetten en bijvoorbeeld automatisch extra chunks ophalen en als de evaluatie nog steeds niet voldoende is, dan kun je de gebruiker daarover informeren.

Ook kun je dit soort gevallen flaggen voor review door een expert, zodat het systeem verbeterd kan worden.

Human-in-the-loop

Afhankelijk van de automatische evaluatie kun je ook de gebruiker automatisch doorschakelen naar een inhoudelijke expert (human).

Algemeen

Leg rationale vast

Leg de rationale van de LLM vast en zorg ervoor dat de **LLM dit weet** (LLM's zijn bewezen nauwkeuriger wanneer ze weten dat ze geëvalueerd worden).

Valkuil = fallback

Een bekende valkuil is dat er een fallback gebouwd wordt wanneer bv. een retrieval-tak niet goed gaat. Echter, dit kan er vaak voor zorgen dat niet de juiste info wordt opgehaald en dus geen juist antwoord gegeven kan worden. De prompt kan dit redelijk tegemoet gaan, maar vraag je af of een fallback helpt, of dat je beter een retry en daarna foutmelding kan geven (beter een foutmelding dan een fout antwoord).

Fine-tuning

Je kunt je modellen fine-tunen voor zijn taak. Let daarbij op dat je niet traint/fine-tuned op informatie die variabel is. Fine-tuning wordt doorgaans vooral gebruikt om de vorm van de uitkomst vooral te verbeteren, niet de inhoud.

Dit, omdat je informatie die je de LLM extra leert weer moet "ontleren" wanneer dit verandert. Dit is een kostbaar proces.

Continue monitoring

Monitor continue de output van de LLM en blijf zoeken naar verbeterpunten. Laat kritieke zaken automatisch flaggen (notificatie), zodat je op tijd bij kritieke problemen bent (bv. als de API van een LLM down is).

Testen

Er zijn verschillende manieren om je systeem te testen.

- Gebruikerstesten / gebruikersfeedback
- A/B-testen
- Adverserial testing (robuustheid testen tegen misbruik/ongewenst gebruik)
- Stress-testing (hoeveel gebruikers kan het aan, welke workload kan het aan)
- Etc.

Eerst klassieke chatbot-antwoord-check

Om zeker te zijn dat antwoorden juist zijn kun je Veelgestelde vragen definiëren. Deze leg je vast in een database met "vraag"-"antwoord". De vraag embed je met je embeddingsmodel. Wanneer een gebruiker van je AI Assistent een vraag stelt, dan wordt eerst die vraag geëmbed en gecheckt met de geëmbedde vragen. Als de score hoger is dan bijvoorbeeld 0.9 (check of dit de juiste threshold is), dan toon je direct het direct antwoord van de FAQ. Geen LLM komt hier aan te pas, waardoor je 100% geen hallucinatie krijgt (en dit proces is razendsnel, waardoor het niet in de weg zit).