EEN VERANTWOORDE AI-CHATBOT VOOR DE EU AI ACT

VOOR

HU AI Lectoraat

DOOR

Achraf, Emre, Henry
Kai & Xander

Inhoudsopgave

Inleiding

Dit project valt binnen het Responsible Applied Artificial Intelligence (RAAIT) programma en richt zich op de verdere ontwikkeling en optimalisatie van een chatbot. Het doel van deze chatbot is om organisaties te ondersteunen bij het navigeren door de EU AI Act. We streven ernaar om deze tool te transformeren tot een open-source platform waar vragen gesteld kunnen worden aan een chatbot, die op een verantwoorde manier is ontwikkeld. In eerste instantie zal de tool worden ingezet voor onderzoek experimenten door de Hogeschool Utrecht (HU).

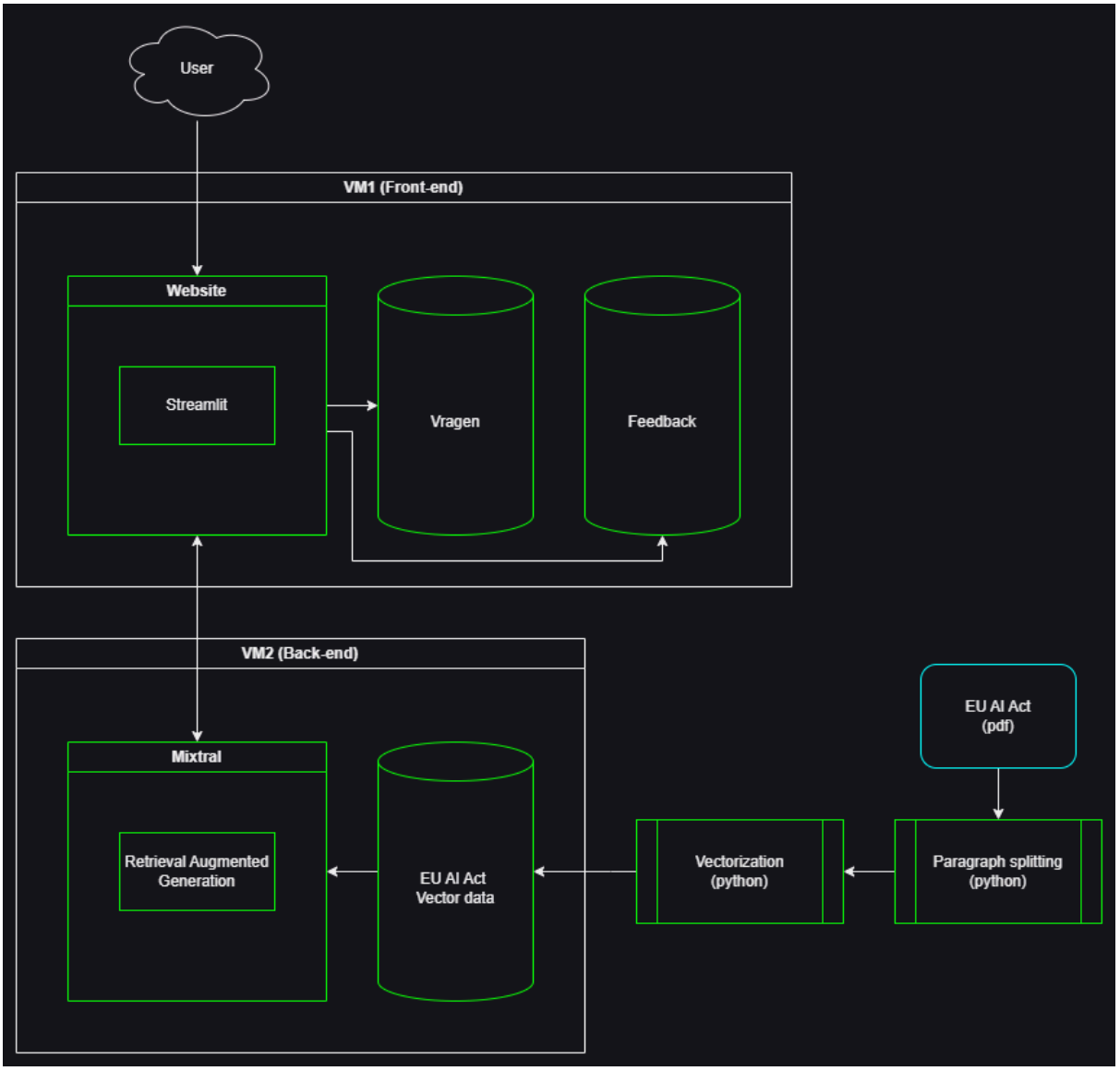
De nadruk van dit project ligt op het verantwoord toepassen van AI, met als centrale vraag: "Hoe zorg je ervoor dat mensen niet blind vertrouwen op een chatbot, terwijl ze hem ook niet onnodig wantrouwen?" We willen dit bereiken door het large language model (LLM) te finetunen op de EU AI Act en door een transparante manier van bronvermelding toe te passen. Hierdoor kunnen gebruikers zien waar de chatbot zijn antwoorden op baseert.

De opdrachtgever wil een discussie-tool in de vorm van een chatbot beschikbaar stellen op de website van Responsible Applied Artificial Intelligence (RAAIT). Dit prototype richt zich op het ontsluiten van de EU AI Act om bedrijfseigenaren te helpen deze wetgeving te begrijpen. Daarnaast willen we onderzoeken hoe LLM's informatie aan leken kunnen verstrekken.
De studenten moeten het Mixtrall 7B model implementeren in een veilige omgeving, waarbij gebruikersfeedback wordt verzameld en gegevens veilig worden opgeslagen.

Het eindproduct moet een LLM zijn dat goed werkt met Retrieval Augmented Generation (RAG), veilig online gebruikt kan worden, en onderhouden kan worden door iemand met basiskennis van programmeren. Het moet direct bruikbaar zijn voor experimenten, maar niet voor serieus advies aan bedrijven. Het is essentieel dat gebruikers zich ervan bewust zijn dat ze een testtool gebruiken en dat het advies niet bedoeld is voor daadwerkelijke toepassing binnen hun bedrijf.

Omgevingstructuur

(mappen structuur uitgelegd, design van de oplossing (die afb van xander)
Link naar de github



Het LLM

Het Large Language Model (LLM) die in dit project wordt gebruikt, is het Mixtral 8x7b-v0.1Q5_K_M model. Dit model is geselecteerd vanwege de voorkeur voor een mixtral model en de hoeveelheid VRam dat dit model nodig is om op een A10 te draaien.

Het Mixtral model wordt geïmplementeerd in een veilige omgeving en is toegankelijk via een webinterface. De backend van deze applicatie is gebouwd met Flask, een lichtgewicht webframework voor Python. Hier is een overzicht van de implementatie:

- Initialisatie: Het model wordt geïnitialiseerd bij het opstarten van de server. Dit omvat het laden van het modelbestand en configureren van parameters zoals de maximale sequentielengte (n_ctx) en het aantal GPU layers (-1 dus het hele model).
- Prompt constructie: Bij ontvangst van een gebruikersvraag vanuit de website, wordt een prompt samengesteld. Deze prompt kan context bevatten die is verkregen via RAG. Dit helpt het model om relevantere en accuratere antwoorden te genereren.
- Generatie van antwoorden: het model genereert antwoorden op basis van de prompt. De gegenereerde output wordt dan teruggestuurd naar de webstie.

Retrieval Augmented Generation (RAG)

Wat is RAG?

Retrieval Augmented Generation (RAG) is een innovatieve techniek waarbij een large language model (LLM) niet alleen afhankelijk is van de kennis uit zijn trainingsdata, maar ook toegang heeft tot relevante externe documenten of realtime data. Hoewel een LLM, zoals ChatGPT, is getraind op een enorme hoeveelheid tekstdata, heeft het geen toegang meer tot die specifieke trainingsdata zodra het is getraind. Dit betekent dat als je ChatGPT iets vraagt over recente gebeurtenissen, zoals het nieuws van gisteren, hij moet aangeven dat zijn trainingsdata niet up-to-date is.

RAG biedt een oplossing door de LLM te verbinden met externe bronnen om relevante context toe te voegen aan de prompts. Dit maakt het mogelijk om actuele en nauwkeurige antwoorden te genereren, gebaseerd op de meest recente informatie. Bovendien kunnen de gebruikte bronnen worden vermeld, wat zorgt voor transparantie en betrouwbaarheid.

In dit project hebben we RAG verbonden met het EU-AI-Act rechtsdocument. Dit stelt de LLM in staat om juridische vragen te beantwoorden met behulp van actuele en specifieke informatie uit dit document. Hierdoor kunnen we niet alleen relevante en nauwkeurige antwoorden genereren, maar ook duidelijke bronvermeldingen geven die aangeven op welke informatie het antwoord is gebaseerd

Hoe is rag toegepast?

RAG bestaat uit meerdere stappen, deze zijn uitgebreid uitgelegd in de notebook, maar samengevat is de notebook een pipeline die van een gegeven input string een relevant stuk tekst op haalt.

Om de tekst te kunnen vergelijken moet de tekst omgezet worden naar een numerieke representatie. Dit heet embedden. Een embedding is een vector die ook semantische relaties in een zin of alinea representeert. De embeddings techniek die wordt gebruikt heet SPLADE. Dit is een sparse embedding techniek. We hebben voor SPLADE gekozen omdat een geschikte techniek is voor het embedden van zinnen en paragrafen. Sparse embedding legt de nadruk op specifieke informatie in een zin of alinea (*Mastering RAG: How to Select an Embedding Model*, 2024). Dit is handig voor de onze use case van RAG omdat je onderwerpen en thema's uit een zin wilt vergelijken met de onderwerpen en thema's uit een alinea in de vector database.

Nu we weten wat embedden is, kunnen we verdergaan met de vector database. In ons geval gebruiken we een JSON-bestand als vector database. Dit bestand bevat alle alinea's van de EU-AI-Act en hun bijbehorende embeddings, die zijn gemaakt met SPLADE.

Het JSON-bestand wordt gebruikt om relevante tekst op te halen. Dit doen we door de ge-embedde gebruikersinput te vergelijken met alle embeddings in het bestand. Voor deze vergelijking gebruiken we de techniek van cosine similarity. De tekst die hoort bij de embedding met de hoogste score wordt vervolgens teruggegeven.

Validatie van RAG

In de notebook voeren we verschillende testcases uit om de robuustheid en performantie van RAG te evalueren. We richten ons specifiek op het gedrag van de functie die de meest vergelijkbare zinnen identificeert op basis van een gegeven vraag. Hierbij worden de volgende testcases behandeld:

Ongerelateerde Vraag: We testen hoe ons systeem omgaat met een vraag die weinig of geen overeenkomst heeft met de opgeslagen zinnen. Dit helpt ons te begrijpen of het systeem in staat is om onderscheid te maken tussen relevante en niet-relevante input.

Ongerelateerd Woord als Input: We onderzoeken hoe ons systeem reageert wanneer een enkelvoudig, ongerelateerd woord als input wordt gegeven in plaats van een volledige vraag. Dit testgeval helpt ons de capaciteit van het systeem te beoordelen om met onvolledige of onduidelijke input om te gaan.

Reeks Gerelateerde Woorden als Input: We stellen een reeks gerelateerde woorden als input voor. Dit scenario simuleert een situatie waarin de vraag complexer is en meerdere aspecten of contextuele informatie bevat. We evalueren of ons systeem in staat is om de juiste zinnen te identificeren die overeenkomen met de gegeven context.

Door verschillende types van input te testen, kunnen we beoordelen of ons systeem consistent blijft in het identificeren van relevante zinnen en het negeren van irrelevante informatie.

Uit deze tests bleek dat het systeem goed onderscheid kon maken tussen relevante en irrelevante onderwerpen. Het systeem gaf namelijk lage similarity scores bij ongerelateerde input, en hoge bij gerelateerde input.

Integratie van rag?

De gebruikte methode van RAG voor dit project is relatief simpel. De EU AI Act is opgesplitst in losse delen, ieder deel is vervolgens omgezet naar een vector embedding. Deze embedding is een soort betekenis van de tekst beschreven in een vector. De embedding en de tekst waarmee deze gemaakt is worden opgeslagen, door vervolgens de ingevulde vraag van de gebruiker ook om te zetten naar zo'n embedding kunnen we in de lijst van de EU AI Act embeddings gaan zoeken naar de meest overeenkomende. Dit doen we door middel van cosine similarity en hiermee krijgen we dus de vector die het 'dichtstbij' de vector van de vraag van de gebruiker zit. Hierna voegen we niet de vector embedding maar het originele stuk tekst van de EU AI Act toe aan de prompt middels een stuk prompt engineering. Hierdoor heeft de LLM meer context over de EU AI Act.

Advies en conclusie

Uit de tests blijkt dat het systeem goed in staat is om onderscheid te maken tussen relevante en irrelevante onderwerpen. Het systeem genereert lage similarity scores voor ongerelateerde input en hoge scores voor gerelateerde input.

1. Uitbreiden van de Testsample: Voor een robuustere evaluatie raden we aan om een grotere testsample te gebruiken. Dit helpt bij het vaststellen van een geschikte drempelwaarde voor de similarity scores.

2. Toevoegen van een Threshold: Na het uitbreiden van de testsample is het aan te raden om een threshold in te stellen. Hierdoor haalt RAG alleen tekst op die voldoende relevant is. Dit kan worden bereikt door een statistische analyse van de scores, waardoor beter kan worden bepaald wat als "relevant" moet worden beschouwd.

Website interface

Waarom streamlit?

De opdrachtgever gaf aan het begin van het project al aan dat de website niet het belangrijkste onderdeel is van dit project, en dat daar dus niet het meeste tijd in moet gaan zitten. Wel wilde hij dat het er een beetje strak uit zag. Met deze kennis en eisen zijn we op zoek gegaan naar een tool die daar aan voldoet. Toen kwamen we al gauw terecht bij de streamlit library en daar is onze keuze uiteindelijk opgevallen. Hij noemde een uitgebreidere website, die bijvoorbeeld gemaakt is met HTML of FLASK, een nice to have.

Functionaliteiten

De website is opgedeeld in pagina's, de hoofdpagina 'home' bevat de chat met de bot. De overige pagina's zijn te vinden in de pages folder, dit zijn: 'feedback', 'disclaimer' en 'contact'. Alle pagina's zijn te vinden in de sidebar van de website en er kunnen nieuwe pagina's aan worden toegevoegd door deze in de pages folder te stoppen. De 2 plekken die verbinden met externe servers/databases (home: chatbot & feedback) beschikken over een asynchronische functionaliteit die op de website aangegeven wordt middels een spinner.

Databases

Feedback database

In de feedback database kan gebruikers feedback worden opgeslagen, deze tabel is gemaakt volgens de standaard auto_increment id van sql. De feedback van de gebruiker bestaat uit een meening over de response time (1-10), de kwaliteit van het antwoord (1-10) en de feedback tekst. Ook is er een timestamp column toegevoegd die bijhoudt wanneer de feedback aan de tabel is toegevoegd.

Field	Type	Null	Key	Default	Extra
id	int	NO	PRI	NULL	auto_increment
response_time	int	NO		NULL	
answer_quality	int	NO		NULL	
feedback	text	YES		NULL	
timestamp	timestamp	YES		CURRENT_TIMESTAMP	DEFAULT_GENERATED

```
drop table if exists feedback;
create table feedback(
  id int primary key auto_increment,
  response_time int not null,
  answer_quality int not null,
  feedback TEXT,
  timestamp timestamp default current_timestamp
);
```

Vragen database

In de vragen database kunnen de gebruikers vragen worden opgeslagen, deze tabel is gemaakt volgens de standaard auto_increment id van sql. De gebruikers vraag bestaat simpelweg uit een vraagtekst. Ook is er een timestamp column toegevoegd die bijhoud waneer de vraag aan de tabel is toegevoegd.

Field	Type	Null	Key	Default	Extra
id	int	NO	PRI	NULL	auto_increment
vraag	text	NO		NULL	
timestamp	timestamp	YES		CURRENT_TIMESTAMP	DEFAULT_GENERATED

```
drop table if exists vragen;
create table vragen(
  id int primary key auto_increment,
  vraag TEXT not null,
  timestamp timestamp default current_timestamp
);
```

Security

Voor de ontwikkeling van onze AI-chatbot hebben we veel aandacht besteed aan beveiliging, om te zorgen dat de gegevens van gebruikers veilig worden verwerkt en opgeslagen. In dit gedeelte van het verslag zullen we uitleggen welke beveiligingsmaatregelen we hebben getroffen en waarom we voor deze opties hebben gekozen.

Gebruik van RestAPI en Private Network
Een belangrijk aspect van de beveiliging van onze AI-chatbot is het gebruik van een RestAPI. RestAPI's zijn een populaire manier om verschillende onderdelen van een applicatie met elkaar te laten communiceren. In ons project zorgt de RestAPI ervoor dat gegevens veilig en efficiënt tussen de chatbot en de backend-services worden verzonden.

Daarnaast hebben we ervoor gekozen om een private network te gebruiken binnen de virtuele machines (VM's) die draaien in de cloudomgeving van Surf Research Cloud. Een private network zorgt ervoor dat de communicatie tussen de verschillende componenten van ons systeem afgeschermd is van het publieke internet. Dit betekent dat alleen geautoriseerde systemen binnen hetzelfde private network toegang hebben tot de gegevens, wat het risico op ongeautoriseerde toegang aanzienlijk vermindert.

Conclusie & Advies

Uit ons project blijkt dat de implementatie van een AI-chatbot, gefinetuned op de EU AI Act en ondersteund door Retrieval Augmented Generation (RAG), een veelbelovende methode is om juridische informatie toegankelijker te maken voor bedrijfseigenaren. Hoewel het doel van het project vooral was om een proefballon op te laten voor het AI-lectoraat, hebben we aangetoond dat ons systeem goed onderscheid kan maken tussen relevante en irrelevante vragen, wat resulteert in nauwkeurige en betrouwbare antwoorden. De transparante manier van bronvermelding draagt bij aan het verantwoord gebruik van AI, doordat gebruikers kunnen zien waar de antwoorden op gebaseerd zijn. Alle componenten werken en het systeem lijkt ons een goede basis als onderzoekstool.

Conclusies

1. **Effectiviteit van RAG**: Onze tests bevestigen dat het gebruik van RAG in combinatie met het Mixtral model effectieve resultaten oplevert. Het systeem kon relevante passages uit de EU AI Act identificeren en gebruiken voor het genereren van nauwkeurige antwoorden.
2. **Gebruiksvriendelijkheid**: De webinterface, gebouwd met Streamlit, biedt een eenvoudige en intuïtieve manier voor gebruikers om interactie aan te gaan met de chatbot. Dit draagt bij aan de toegankelijkheid van de tool, zelfs voor gebruikers met beperkte technische kennis.
3. **Veiligheid en Privacy**: De toepassing van beveiligingsmaatregelen, zoals het gebruik van een RestAPI en een private network binnen de Surf Research Cloud, zorgt voor de veilige verwerking en opslag van gebruikersgegevens. Dit is cruciaal voor het waarborgen van de privacy en het vertrouwen van de gebruikers.
4. **Transparantie**: Het toevoegen van bronvermeldingen aan de gegenereerde antwoorden verhoogt de transparantie van de chatbot. Gebruikers kunnen zien waarop de antwoorden zijn gebaseerd, wat bijdraagt aan een verantwoorde toepassing van AI.

Aanbevelingen

1. **Uitbreiding van de Test Sample:** Voor een robuustere evaluatie van het systeem raden we aan om een grotere test sample te gebruiken. Dit helpt bij het nauwkeuriger vaststellen van drempelwaarden voor similarity scores, waardoor de relevantie van opgehaalde teksten verder kan worden verbeterd.
2. **Instellen van een Threshold**: Na het uitbreiden van de test sample is het aan te raden een threshold in te stellen. Hierdoor zal RAG alleen teksten ophalen die voldoende relevant zijn. Dit kan worden bereikt door middel van een statistische analyse van de similarity scores.
3. **Gebruikersfeedback**: Het verzamelen en analyseren van gebruikersfeedback is essentieel voor de doorlopende verbetering van de chatbot. Door regelmatig feedback te evalueren, kunnen zwakke punten in het systeem worden geïdentificeerd en aangepakt. Ook biedt dit inzicht in het gebruik van de chatbot en dat is relevant voor het onderzoek.
4. **Educatie van gebruikers:** Het is belangrijk dat gebruikers zich bewust zijn van de beperkingen van de chatbot. We raden aan om duidelijke disclaimers te communiceren en gebruikers te informeren dat de tool bedoeld is voor testdoeleinden en niet voor juridisch advies.
5. **Onderhoud en updates:** Het onderhouden van de chatbot en het regelmatig bijwerken van de juridische informatie zijn cruciaal om de relevantie en nauwkeurigheid van de antwoorden te waarborgen. Dit vereist een plan voor doorlopende support, updates en het bijwerken van de RAG database.

Door deze aanbevelingen op te volgen, kan de AI-chatbot verder worden geoptimaliseerd en een waardevol hulpmiddel worden voor bedrijfseigenaren die navigeren door de complexe EU AI Act. Het project toont aan dat het mogelijk is om een verantwoorde en effectieve AI-oplossing te ontwikkelen die bijdraagt aan de toegankelijkheid van juridische informatie.

References

Mastering RAG: How to Select an Embedding Model. (2024, March 5). Galileo. https://www.rungalileo.io/blog/mastering-rag-how-to-select-an-embedding-model