Systémová dokumentace pro RAG Chatbot s využitím LangGraph

Cílem projektu bylo vyvinout interní (in-house) řešení schopné simulovat chování populárních online chatbotů, jako jsou například OpenAI ChatGPT nebo Google Gemini. Hlavní výhodou tohoto přístupu je možnost vysoké míry customizace – od úpravy architektury podle potřeb jednotlivých zákazníků až po volbu konkrétních jazykových modelů. Uživatelé si tak mohou natrénovat vlastní model nebo provozovat lokální řešení na on-premise infrastruktuře. Alternativní možností je i nasazení instance OpenAI v prostředí Azure Cloud a její integrace s naším systémem.

Projekt představuje pokročilý RAG (Retrieval-Augmented Generation) systém, který je postaven na frameworku LangGraph. Backendová část zajišťuje orchestraci vyhledávání a generování odpovědí, zatímco klientská aplikace je vyvinuta v React frameworku. Základní architektura kombinuje webové vyhledávání s pokročilou analýzou dokumentů a jako výchozí model využívá Google Gemini prostřednictvím API rozhraní.

Hlavní funkcionality

Konverzační rozhraní

Uživatel může posílat dotazy a přijímat odpovědi, přičemž je podporováno i streamování zpráv (odpovědi se zobrazují postupně v reálném čase).

Webové vyhledávání

Systém generuje vyhledávací dotazy, které jsou odesílány do Google Search API a následně použity pro zpřesnění odpovědi.

Re-ranking výsledků

Získané výsledky z webu jsou automaticky přehodnoceny a seřazeny podle relevance, což zajišťuje vyšší kvalitu výstupů.

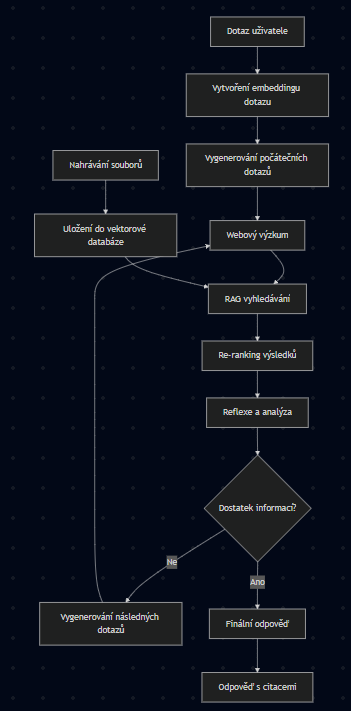
Nahrávání a ukládání souborů

Uživatel může nahrávat soubory ve formátech PDF, Word, PowerPoint, Excel i prosté textové soubory. Tyto dokumenty jsou ukládány do vektorové databáze Pinecone pro následné vyhledávání a analýzu.

Reflexe odpovědí

V předposledním kroku systém provádí hodnocení a případnou korekci vlastních odpovědí, což zvyšuje jejich přesnost a kvalitu.

Proces interakce se systemem



Architektura – [odkaz](https://drive.google.com/file/d/1sc-v_8VEMZaIuIxczCs1M96txz87W1HM/view?usp=sharing)

Použité technologie a architektura

Projekt je postaven na architektuře Retrieval-Augmented Generation systému s využitím frameworku LangGraph, který zajišťuje orchestraci AI workflow. Frontendová část aplikace je vyvinuta v Reactu a je napsána v TypeScriptu. Využívá buildovací nástroj Vite pro rychlý vývoj a nasazení, styling je řešen pomocí Tailwind CSS a knihovny Shadcn UI, která poskytuje předdefinované uživatelské komponenty. Pro komunikaci s backendem slouží HTTP klient Axios a pro správu navigace je integrován React Router.

Backend je implementován v Pythonu (verze 3.11 a vyšší) a používá webový framework FastAPI, který zajišťuje definici API endpointů a správu requestů. Hlavní orchestraci agentů a procesů zajišťuje LangGraph, zatímco knihovna LangChain slouží jako rozšíření pro práci s jazykovými modely a retrieval pipeline. Validace dat a správa datových modelů jsou řešeny pomocí knihovny Pydantic.

V oblasti AI a strojového učení je využit model Google Gemini 2.0 Flash jako hlavní LLM prostřednictvím API rozhraní. Pro embedding modely je nasazena knihovna HuggingFace Transformers a pro re-ranking výsledků jsou využívány Sentence Transformers.

Datová vrstva kombinuje několik typů úložišť. Vektorové reprezentace dokumentů jsou ukládány do Pinecone, který slouží jako vektorová databáze. Redis je použit pro cache, ukládání dočasných výsledků dotazů a také pro real-time komunikaci pomocí pub-sub. Relační databáze PostgreSQL uchovává metadata, historii konverzací a stav agentů. Pro dočasné soubory je využíván lokální souborový systém.

Z hlediska DevOps a infrastruktury je aplikace plně kontejnerizována pomocí Dockeru. Orchestrace více služeb je zajištěna prostřednictvím Docker Compose. Správa Python balíčků je řešena pomocí UV. Pro kontrolu kvality kódu je na frontendové části nasazen ESLint a na backendu nástroj Ruff pro linting a statickou analýzu.