Retrieval Augmented Generation (RAG)

LLm hanno conoscenza limitata solo ai dati usati durante il training, non possono accedere a nuove informazioni introdotte dopo il training, non possono ragionare su dati privati o proprietari.

Un'applicazione RAG ha 2 componenti principali: indexing e retrieval

Indexting: una pipeline per prendere dati da una fonte ed indicizzarla, viene fatto offline

Retrieval and generation: prende la query dell'utente, recupera i dati rilevanti dall'indidice, genera un prompt che contiene la query e i dati recuperati come contesto, manda questo prompt ad una llm per generare una risposta.

Indexing

LOAD: Prima di tutto dobbiamo caricare i dati, possiamo caricare pdf,csv,html e lui li trasforma in embedding.

SPLIT: Divido i documenti in piccoli chunk, non usa tutto il documento perché di solito la risposta è contenuta in una parte del documento, di solito questi chunk si sovrappongono per evitare che l'informazione sia sparsa tra piu chunks.

La context window di un llm è finita quindi mettere grossi documenti potrebbe produrre informazioni non importanti.

STORE: memorizziamo ogni chunk e indice in uno search engine, vector store

VECTOR STORES

Documenti che parlano dello stesso topic sono posizionati vicini in base ai loro embeddings, RICERCA PER SIMILAIRTA'

RETRIEVAL AND GENERATION

Le informazioni rilevanti vengono recuperate, viene genarato un prompt, passato alla rete e viene prodotta la risposta.

INTRODUCTION TO LANGCHAIN

Framework che semplifica lo sviluppo di applicazioni che usano llm.

Sono blocchi per incorporare le llm in applicazioni.

Posso connettere llms di terze parti, data sources e tool esterni.

Vari casi d'uso come chatbots, rag, document search..

Prompt templates: facilita la traslazione di user input in llm instructions

Llms: llm models che prendono e ritornano stringhe

Chat models: mantiene la storia della conversazione

Esample selector: seleziona e crea esempi conctreti per metterli nel prompt e migliorare le performance.

OutputParser: trasforma il tresto in modelli struttura come vsv

Document loaders: carica documenti da varie fonti di dati

Vector stories: sistema per memorizzare e recuperare documenti

Retirevers: intrafaccia per documenti e recupero di dati

Agents: sistema basato su llm per ragionare e decidere l'azione basandosi sull'input. Si basa su step precedenti.

ESERCIZIO

Posso usare LCEL per creare piplein modulari, include anche ChainPredefinite per QUESTION ANSWERING per esempio.

#POTREI USARE UN CHAT MODEL NEL PROGETTO CHE TIENE MEMORIA DELLA CONVERSAZIONE

E USARE LE RISPOSTE PRECEDENTI PER RIFINIRE