M13 - SQL RAG

Stand: 03.2025

1 Intro

Retrieval Augmented Generation (RAG) hat sich als eine revolutionäre Technik im Bereich Natural Language Processing (NLP) und Large Language Models (LLMs) erwiesen. RAG kombiniert traditionelle Sprachmodelle mit einem innovativen Retrieval-Mechanismus, der es den Sprachmodellen ermöglicht, auf eine riesige Wissensbasis zuzugreifen, um die Qualität und Relevanz ihrer Antworten zu verbessern. RAG ist besonders wertvoll, wenn detaillierte und aktuelle Informationen benötigt werden.

Die Landschaft der datengesteuerten Technologien entwickelt sich ständig weiter, und verschiedene Methoden und Techniken werden eingesetzt, um die Leistungsfähigkeit von Daten zu nutzen. Daten können entweder **strukturiert** (z. B. in SQL-Datenbanken) oder **unstrukturiert** (z. B. Textdokumente) sein. RAG-Systeme nutzen diese unterschiedlichen Datenarten, um umfassende Antworten zu generieren.

2 Was ist RAG?

RAG ist der Prozess des **Abrufens** relevanter Kontextinformationen aus einer Datenquelle und des Weitergebens dieser Informationen zusammen mit der Eingabeaufforderung des Benutzers an ein großes Sprachmodell. Diese Informationen werden verwendet, um die Ausgabe des Modells (generierter Text oder Bilder) durch **Erweitern** des Basiswissens des Modells zu **verbessern**.

RAG ist wertvoll für Anwendungsfälle, in denen das Modell Wissen benötigt, das nicht in den Informationen enthalten ist, die das Modell gelernt hat. Wenn Sie beispielsweise einen GPT erstellen, um Ihr Support-Team bei der Beantwortung von Kundenanfragen zu unterstützen, kann GPT-4 zwar Kundenprobleme mithilfe seines Basiswissens bearbeiten, aber nicht die neuesten Fakten zu Ihrem spezifischen Produkt oder Ihrer Dienstleistung kennen. Durch den Zugriff auf Ihr Ticketsystem kann ein GPT vergangene Tickets zu ähnlichen Problemen abrufen und diesen Kontext verwenden, um relevantere Antworten zu generieren.

3 Warum SQL für RAG?

Das Erstellen eines Retrieval-Augmented Generation (RAG)-Systems bringt mehrere Herausforderungen mit sich, aber SQL könnte helfen, diese zu bewältigen:

- **SQL kann helfen, komplexe Daten abzurufen** Das Abrufen relevanter Informationen aus riesigen und vielfältigen Datensätzen kann komplex sein, insbesondere beim Umgang mit unstrukturierten oder semistrukturierten Datenquellen wie Textdokumenten, Bildern oder Multimedia. Die Integration effizienter Retrieval-Mechanismen, die diese Komplexität bewältigen können, ist eine bedeutende Herausforderung. Die Abfragefunktionen von SQL ermöglichen den effizienten Abruf relevanter Informationen aus diesen Datenquellen. Durch das Generieren von SQL-Abfragen, die auf bestimmte Kriterien zugeschnitten sind, und die Nutzung erweiterter Suchfunktionen kann SQL den Datenabrufprozess optimieren und so die Komplexität des Zugriffs auf verschiedene Datensätze bewältigen.
- **SQL kann helfen, Qualitätsdaten abzurufen** Die Sicherstellung der Qualität und Relevanz der abgerufenen Daten ist entscheidend für die Generierung genauer und sinnvoller Antworten. Verrauschte oder veraltete Daten sowie irrelevante Informationen können die Leistung des RAG-Systems jedoch negativ beeinflussen. Die Entwicklung von Algorithmen zum effektiven Filtern und Ranking abgerufener Daten ist eine Herausforderung. SQL bietet Mechanismen zum Filtern und Ranking abgerufener Daten basierend auf verschiedenen Kriterien wie Zeitstempeln, Kategorien oder Relevanzwerten.
- **SQL bietet Skalierbarkeit und Flexibilität** Da Datensätze an Größe und Komplexität zunehmen, wird Skalierbarkeit zu einer großen Herausforderung für RAG-Systeme. Die Sicherstellung, dass das System mit zunehmenden Datenmengen umgehen kann und gleichzeitig Leistung und Reaktionsfähigkeit aufrechterhält, erfordert ein effizientes Architekturdesign und Optimierungsstrategien. SQL-Datenbanken sind darauf ausgelegt, riesige Mengen strukturierter Daten effizient zu verwalten. Die Integration von SQL in RAG-Systeme adressiert eine der wichtigsten Herausforderungen im Bereich der KI: die Skalierung des Retrieval-Mechanismus zur Handhabung umfangreicher Datensätze, ohne die Leistung zu beeinträchtigen. Darüber hinaus ermöglicht die Flexibilität von SQL bei der Formulierung von Abfragen RAG, komplexe Informationen abzurufen und dabei die Breite und Tiefe der während des Generierungsprozesses berücksichtigten Daten anzupassen.
- **SQL hilft beim Abrufen von Echtzeitdaten** Die Bereitstellung von Echtzeitantworten ist für viele Anwendungen von RAG-Systemen, wie z. B. Chatbots oder virtuelle Assistenten, von entscheidender Bedeutung. Das Erreichen niedriger Latenzzeiten bei gleichzeitiger Aufrechterhaltung der Qualität der generierten Inhalte stellt eine Herausforderung dar, insbesondere in Szenarien mit strengen Latenzanforderungen. Die Optimierungstechniken von SQL, wie z. B. Query-Caching und Indizierung, können die

Query-Verarbeitungszeiten erheblich reduzieren und es RAG-Systemen ermöglichen, Echtzeitantworten bereitzustellen.

4 RAG mit SQL-Workflow

Der LLM wandelt die Frage in natürlicher Sprache in eine SQL-Abfrage um, die in der SQL-Datenbank ausgeführt wird. Die Datenbank gibt Datensätze zurück, die dann vom LLM verarbeitet werden, um die endgültige Antwort zu generieren.

5 Praktische Schritte zur Integration von RAG mit Ihrer SQL-Datenbank

- **Bewerten Sie die Datenqualität**: Beginnen Sie mit der Bewertung der Qualität und Relevanz Ihrer Quelldaten und stellen Sie sicher, dass sie die Standards für effektive RAG-Interaktionen erfüllen.
- **Modellintegration**: Integrieren Sie das RAG-Modell in Ihre SQL-Datenbank und stellen Sie Verbindungen für den Echtzeit-Datenabruf und die -Analyse her.
- **JavaScript-Interaktion**: Entwickeln Sie JavaScript-Schnittstellen, die Benutzerinteraktionen mit dem integrierten RAG-System ermöglichen und natürliche Sprachabfragen ohne direkte SQL-Beteiligung ermöglichen.

6 Tipps und Tricks zur Beherrschung der RAG-SQL-Integration

Um die Leistung Ihrer RAG-SQL-Integration zu verbessern, sollten Sie die Implementierung von Caching-Mechanismen in Betracht ziehen, um häufig aufgerufene Daten vorübergehend zu speichern. Durch das Cachen von Abfrageergebnissen können Sie die Antwortzeiten verkürzen und die Gesamteffizienz des Systems verbessern. Darüber hinaus kann die Optimierung der Indizierung für wichtige Spalten innerhalb Ihrer Datenbank die Datenabrufprozesse erheblich beschleunigen und schnelle Antworten auf Benutzerabfragen gewährleisten.

7 Lokale SQL-Generierungsarchitektur

Die vorgeschlagene Architektur unterscheidet sich grundlegend von den herkömmlichen Retrieval-Augmented Generation (RAG)-Modellen. Anstatt sich auf Embedding-Vektoren zu verlassen, um Informationen abzurufen, dreht sich unser Modell um die Fähigkeit des LLM, Benutzerabfragen zu interpretieren und in SQL-Befehle zu übersetzen. Diese Befehle werden auf dem lokalen Gerät des Benutzers ausgeführt, und nur die bereinigten Ergebnisse werden an das LLM zurückgegeben.

1. **Benutzerabfrageeingabe:** Die Reise beginnt mit einer Benutzerabfrage in natürlicher Sprache, die von einfachen Anfragen bis hin zu komplexen Anweisungen für die

Datenbankinteraktion reicht.

2. **LLM als Vermittler:** Das LLM tritt ein, nicht als Datenverarbeiter, sondern als Übersetzer, der die natürliche Sprachabfrage in einen SQL-Befehl umwandelt. Das Modell ist darauf ausgelegt, die Absicht und die erforderlichen Datenoperationen zu verstehen, ohne auf tatsächliche Daten zugreifen zu müssen.
3. **Ausführung des SQL-Befehls lokal:** Der generierte SQL-Befehl wird in der lokalen Umgebung des Benutzers ausgeführt – sei es eine Cloud-basierte Datenbank oder eine lokale Browserdatenbank (Alasql). Diese lokale Ausführung stärkt die Sicherheit der Benutzerdaten, indem sie innerhalb der vom Benutzer gewählten Umgebung gehalten werden.
4. **Lokale Daten zu informativer Antwort:** Nach der Ausführung empfängt das LLM die Ergebnisse, die es dann verwendet, um eine informative und benutzerfreundliche Antwort zu formulieren.
5. **Kontinuierliches Lernen:** Das Modell ist darauf ausgelegt, aus jeder Interaktion zu lernen und das Benutzerfeedback zu nutzen, um die Genauigkeit der SQL-Übersetzung zu verfeinern, ohne persönliche Daten zu speichern.