Využitie LLM pre analýzu právnych dokumentov: Hybrid GraphRAG Documentation

Samuel Bagín

February 3, 2026

Contents

1 Úvod

S nárastom využívania veľkých jazykových modelov (LLM) v rôznych oblastiach sa objavuje potreba efektívnych metód na analýzu a spracovanie špecifických typov dokumentov, ako sú právne dokumenty. Samotné LLM si často nevedia poradiť s komplexnými štruktúrami a vzťahmi v textoch, čo vedie k halucinácií a obmedzeniam v ich schopnosti poskytovať presné a relevantné informácie.

Veľké jazykové modely uľahčujú extrakciu entít a vzťahov z textu. Alternatívne riešienie na extrakciu entít a vzťahov by bolo možné využiť strojové učenie, avšak to by vyžadovalo trénovanie modelov na určený jazyk, extrakciu presne stanovenej ontológie a využitie veľkého množstva dát na efektívne natrénovanie modelu. Slovenská legislatívna doména je veľmi veľká, komplexná a členitá.

Ako riešenie prichádza využitie LLM na transformáciu neštruktúrovaného textu na štruktúrovné dáta. Uloženie dát do znalostného grafu (KG) a následné využitie týchto dát na zodpovedanie otázok pomocou metódy GraphRAG (Graph Retrieval-Augmented Generation).

2 Cieľ práce

Cieľom projektu je vytvoriť AI systém na automatickú extrakciu a prepojenie informácií z právnych textov do znalostného grafu s pokročilým sémantickým vyhľadávaním. Systém kombinuje grafovú databázu s vektorovým úložiskom pre hybridné vyhľadávanie, ktoré umožňuje používateľom klásť otázky v prirodzenom jazyku a získavať presné odpovede na základe štruktúrovaných vzťahov aj sémantickej podobnosti.

3 System Overview

Tento projekt je postavený na pomocou jazyku Python a využíva knižnicu LangChain.

3.1 Využívané LLM

- OpenAI
  - **gpt-4o**: Model používaný na extrakciu schémy, entít a vzťahov z textu. Najbohatšia a najpresnejšia extrakcia. Cena (I/O pre 1M tokenov): **$2.50** / **$10.00**
- Anthropic
  - **sonnet-3.5**: Model používaný na tvorenie Cypher Queries pre dotazovanie sa na grafovú databázu. Silné znalosti na tvorenie kódu a SQL, rýchly, efektívny a riešenie problémov. Cena (I/O pre 1M tokenov): **$3.00** / **$15.00**
- Google
  - **gemini-3-flash**: Model používaný na klasifikáciu dokumentu a vytvorenie odpovede na základe získaných dát. Najlepšie pre spracovanie a zosumarizovanie veľkých kontextových okien. Cena (I/O pre 1M tokenov): **$0.50** / **$3.00**

3.2 Databázy

Pre používanie tohto projektu, používateľ si musí stiahnuť a nainštalovať aplikáciu Neo4j Desktop, a vytvoriť si lokálnu inštanciu.

- Neo4j: Grafová databáza na ukladanie entít a vzťahov.
- Neo4jVector: Neo4j vektorová databáza na ukladanie vektorových embeddingov (vzťahy a entity) a vykonávanie vektorového vyhľadávania.

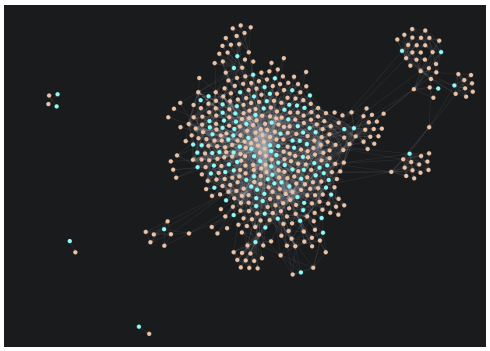

Figure 1: Ukážka grafovej databázy

3.3 Embeddingy

Na embeddovanie chunkov (rozsekané častí textu dokumentu), entít a vzťahov na vektory, používam od OpenAIEmbeddings model **text-embedding-3-large**. Tieto embeddingy (iba vzťahy a entity) sú uložené v Neo4jVector databáze a slúžia na hybridné rýchlejšie vyhľadávanie. Na základe sémantickej podobnosti sú vrátené entity a vzťahy z položenej používateľovej otázky a poskytnuté LLM na dopytovanie znalostného grafu.

3.4 Návod na používanie

Po tom čo si používateľ nainštaluje Neo4j Desktop, musí si vytvoríť lokálnu inštanciu, zapamätať si meno (väčšinou `neo4j`), heslo a uri (väčšinou `neo4j://127.0.0.1:7687`). Následne si treba stiahnuť knižnice z requirements.txt. Do vytvoreného súboru **.env**, používateľ zadá svoje API kľúče pre OpenAI, Anthropic, Google a Neo4j.

```
NEO4J_URI=your_neo4j_uri_here
NEO4J_USERNAME=neo4j
NEO4J_PASSWORD=your_password_here
GOOGLE_API_KEY=your_google_api_key_here
OPENAI_API_KEY=your_openai_api_key_here
ANTHROPIC_API_KEY=your_anthropic_api_key_here
```

Následne, pre používanie treba zadefinovať v **main.py**:

```python
import os
from dotenv import load_dotenv
from pdf_graphrag import PDFGraphRAG

load_dotenv()

graphrag = PDFGraphRAG(
    neo4j_uri=os.getenv("NEO4J_URI"),
    neo4j_user=os.getenv("NEO4J_USERNAME"),
    neo4j_password=os.getenv("NEO4J_PASSWORD"),
    openai_api_key=os.getenv("OPENAI_API_KEY"),
    google_api_key=os.getenv("GOOGLE_API_KEY"),
    claude_api_key=os.getenv("ANTHROPIC_API_KEY"),
    vector_store_nodes_name='node_store',
    vector_store_relationships_name='relationship_store'
)
```

4 Spracovanie