Optimizarea Sistemelor RAG prin Metadate Semantice și Structurare Ontologică

Implementare: Semantic-RAG-LangExtract

Telu Mihai Fugulin Victor

Azure OpenAI & Azure AI Search Implementation

FMI - Tehnici Cloud Computing pentru ML si GenAI

Agenda

Context și Provocări în RAG Standard

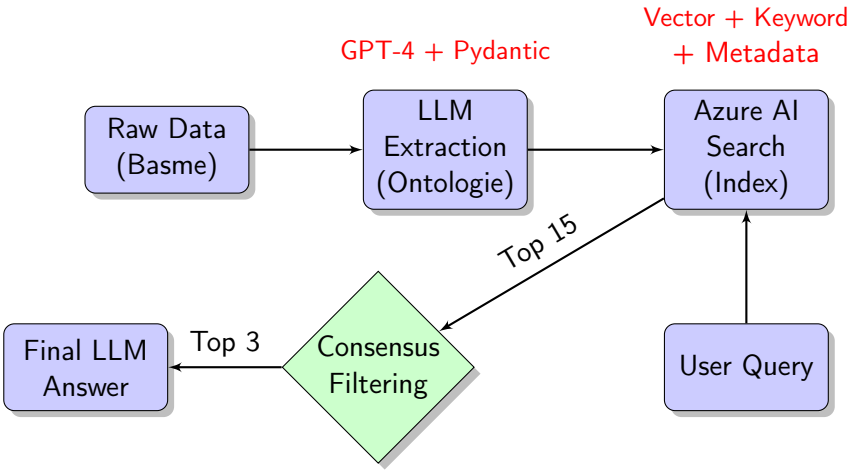
Retrieval-Augmented Generation (RAG) este standardul actual pentru a oferi context LLM-urilor, dar întâmpină limitări:

- **Ambiguitate Semantică:** Căutarea vectorială poate aduce documente similare matematic, dar irelevante contextual (ex: "lupul" din *Scufița Roșie* vs. "lupul" din altă poveste).
- **Pierderea Contextului:** Spargerea textului în "chunks" (fragmentare) rupe legătura narativă (cine vorbește? care este emoția?).
- **Zgomot în Retrieval:** Un top-K (ex: 3 documente) bazat strict pe vectori poate introduce informații contradictorii, generând halucinații.

Obiectivul Proiectului (Cerința 8)

Optimizarea RAG prin extragerea de **Metadate Semantice** și aplicarea unei **Ontologii** pentru filtrarea post-retrieval.

Pipeline-ul Semantic RAG



Corpus de Test: Basme Clasice

Sistemul a fost testat pe un set de date narativ complex, predispus la confuzii de context.

Fișiere procesate (Raw Data Lake):

- *Aladdin and the Wonderful Lamp*
- *Cinderella*
- *Hansel and Gretel*
- *Little Red Riding Hood*
- *Little Snow-White*
- *The Snow Queen*
- ...și alte povești cu structură similară.

Provocare: Întrebarea "Cine este personajul negativ?" poate returna răspunsuri din mai multe povești simultan dacă nu există filtrare semantică.

Etapa 1: Ingestia și Extragerea Semanticii

Fiecare fragment de text (chunk de 1000 caractere) este trecut printr-un LLM (‘GPT-4.1‘) pentru a extrage o structură definită strict prin **Pydantic**.

```python
class SemanticMetadata(BaseModel):
    """Metadate semantice extrase pentru optimizare RAG."""

    summary: Optional[str] = Field(..., description="Rezumat concis.")

    characters: List[str] = Field(..., description="Personaje.")

    emotions: List[str] = Field(..., description="Emotii.")

    topics: List[str] = Field(..., description="Concepte cheie.")
```

Rezultat: Indexul nu conține doar text, ci și *sensul* structurat al acestuia.

Etapa 2: Consensus Filtering (Post-Retrieval)

Diferența majoră față de RAG-ul clasic este aplicarea unui algoritm de **Coeziune Semantică** asupra rezultatelor brute (Top 15).

Algoritmul de Scoring:

1. **Analiza Frecvenței:** Se calculează frecvența apariției personajelor și topicurilor în toate cele 15 documente returnate.
2. **Punctare (Scoring):** Fiecare document primește un scor de coeziune:

$$S_{doc} = \sum(T_{common} \times 2) + \sum(C_{common} \times 3) + \sum(E_{common} \times 1)$$

   Unde C_{common} sunt personaje care apar în cel puțin alte 2 documente din set.
3. **Re-ranking:** Documentele sunt reordonate descrescător după scorul de coeziune.
4. **Selecția:** Se păstrează doar Top 3 cele mai coerente documente.

Implementare: Calculul Scorului

Fragment din `chat_with_metadate.py` care elimină "zgomotul":

```python
    # Calcul scor coeziune
    scored_docs = []
    for doc in retrieved_docs:
        score = 0
        # +2 puncte pentru topicuri comune
        for t in doc.metadata.get("topics", []):
            if topic_counts[t] >= 2: score += 2

        # +3 puncte pentru personaje comune (pondere mare)
        for c in doc.metadata.get("characters", []):
            if char_counts[c] >= 2: score += 3

        scored_docs.append((doc, score))

    # Sortare si Selectie Finala
    scored_docs.sort(key=lambda x: x[1], reverse=True)
    selected_docs = [doc for doc, score in scored_docs[:3]]
```

Etapa 3: Construcția Contextului Îmbogățit

LLM-ul final nu primește doar textul brut ("Page Content"), ci un bloc de informații structurat, ajutându-l să înțeleagă atmosfera și personajele înainte de a citi textul propriu-zis.

Structura Prompt-ului:

```
--- FRAGMENT ---
SUMMARY: Aladdin finds a lamp in a cave...
KEY CHARACTERS: Aladdin, Magician
PREDOMINANT EMOTIONS: Fear, Curiosity
PREDOMINANT TOPICS: Magic, Deception
CONTENT:
"The earth trembled a little and opened in front of them..."
```

Acest **Prompt Engineering** dinamic reduce drastic riscul de confabulare, deoarece modelul este forțat să respecte metadatele validate.

Studiu Comparativ: Standard vs. Semantic RAG

RAG Standard (Fără Metadate)	**Semantic RAG (Cu Metadate)**
Se bazează doar pe similaritate vectorială și cuvinte cheie.	Include un strat de validare semantică (ontologie).
Risc mare de a amesteca fragmente din povești diferite.	Filtrează documentele "outlier" care nu au personajele majoritare din context.
Contextul trimis la LLM este doar text brut.	Contextul este îmbogățit cu rezumate și emoții.
Rezultat: Răspunsuri corecte factual, dar uneori inconsistente narativ.	**Rezultat:** Răspunsuri precise, coerente și ancourate în firul narativ corect.

Concluzii

- **Validarea Cerinței:** Proiectul demonstrează cu succes integrarea metadatelor semantice într-un pipeline RAG.
- **Inovația:** Utilizarea *Consensus Filtering* permite sistemului să se "auto-corecteze" înainte de a genera răspunsul, eliminând fragmentele care, deși similare vectorial, aparțin unui alt context narativ.
- **Scalabilitate:** Arhitectura bazată pe Azure (Data Lake, AI Search, OpenAI) este robustă și poate fi extinsă pentru domenii tehnice, juridice sau medicale, nu doar literare.

Întrebări?