Chat with Financial Documents

Intro to Retrieval Augmented Generation and LangChain

Jan 18, Yifei Gu

Objective

- Build a simple RAG project that you can write on your resume
    - Introduce embedding concept
    - Introduce vector search
    - Explore LangChain, OpenAI API and various embedding models

Limitations of LLMs

Current state of LLMs suffers from:

- Knowledge cut-offs due to the nature of training process
- Bias from training data
- Hallucination – Giving out “made up” answers ungrounded

Solutions:

- Fallback strategy: answer “I don’t know”
- Providing more capabilities to the model:
    - Provide grounded text relevant to the query to the model “in the prompt”
        - Knowledgebase
        - Browsing

Retrieval Augmented Generation

Retrieval-Augmented Generation (RAG) is a technique that combines the power of language models with external knowledge sources to enhance the quality and relevance of generated text.

1. **Retrieval component**: When a query is given to the system, the retrieval component searches a large database of documents or data to find relevant information.
2. **Augmentation:** Present the relevant information to LLM through prompt
3. **Integration:** Actively using that information to guide its generation process

Almost every use case of LLMs utilize RAG, think of:

- GPT with browsing
- Chat with PDFs
- Chat with your own data & knowledge
- ...

Embeddings

- A concept to represent words, sentences, documents as **vectors** of real numbers.
    - A word embedding is a real-valued vector representation of a word
- Embeddings capture **semantic** meanings vs traditional NLP techniques (TF-IDF etc.)

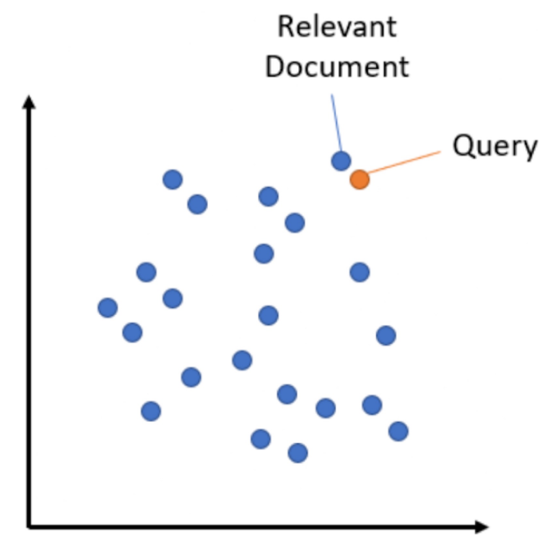
- **Word Embeddings: (Word2Vec, GloVe)**
    - Operate at the word level
    - Capture the semantic meaning of individual words
    - Static: same word has the same vector representation regardless of its context
- **Sentence Embeddings: (BERT, GPT …)**
    - Represent entire sentences or even paragraphs as vectors
    - The overall meaning and semantic content of a sentence or a paragraph
    - Dynamic or contextual**,** the representation of a word can change based on the sentence

Example Scenario: The word "light":

- "He switched on the light to brighten the room.”

- "Her backpack was light, making it easy to carry around."

Vector Search



- Typical Vector Search metrics:
  - Cosine Distance (Equivalent to = 1 - Cosine Similarity):
    - **Cosine Similarity(A, B)** = $\frac{A \cdot B}{||A|| \cdot ||B||}$
      - Range: -1 to 1
      - -1 indicates exactly opposite, 1 indicates exactly the same, 0 means orthogonal;
      - In real world, most models are normalized, so just compute a simple dot product
    - **Cosine Distance(A, B)** = 1 – Cosine Similarity(A, B)
      - Range: 0 to 2
  - Euclidean Distance:
    - **Euclidean Distance(A, B)** = $\sqrt{\sum_{i=1}^{n}(A_i - B_i)^2}$
      - Range: 0 to ∞
      - 0 indicates identical

Vector Databases

Why Vector DB?

- Store large volume of high dimension vector data
- Efficient and fast similarity search
- Scalability

- Typical Vector DB solutions:
    - Pinecone, Weaviate, Chroma, Qdrant
    - Existing DB solutions that support vector search: PostgreSQL, etc.
    - Works out of the box: FAISS

Chat with Financial Documents

`https://github.com/g-1f/workshop`

Steps:

- Install packages and dependencies
- Download SEC documents from instruction
- Add API keys to the environments
- Try out various embedding models, chat models and prompt!
- Chat!