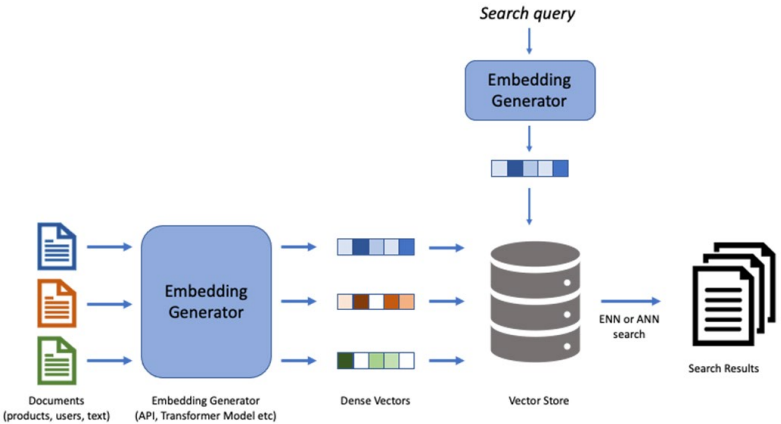
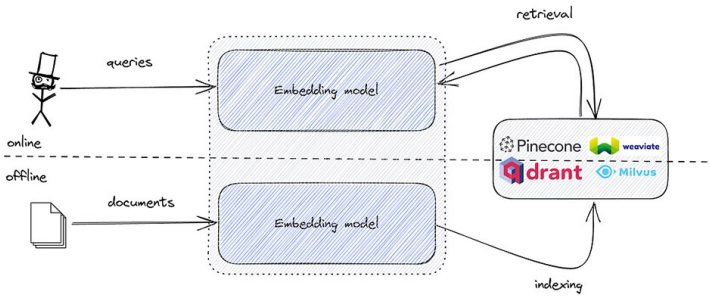
Analytics Vidhya

Mastering LLMs

Day 13: Semantic Search Index



A **Semantic Search Index** is a specialized data structure used to facilitate **semantic search**, which focuses on understanding the meaning and intent behind a user's query rather than relying solely on keyword matching. This type of index allows search engines and applications to return more relevant and context-aware results by leveraging **natural language processing (NLP)**, **machine learning**, and **vector-based search techniques**.



Key Components of a Semantic Search Index

Vector Representations (Embeddings):

- Instead of storing raw text or keywords, semantic search indexes rely on **word embeddings** (e.g., Word2Vec, GloVe, BERT) to convert text into high-dimensional numerical representations. These vectors capture contextual meaning, synonyms, and relationships between words and concepts.
- Example: The words "buy" and "purchase" will have similar vector representations.

Indexing Techniques:

- Traditional keyword-based indexing uses structures like **inverted indexes**, whereas semantic search often leverages **approximate nearest neighbor (ANN)** algorithms to index embeddings efficiently.
- Common indexing methods include:

  - K-D Trees

  - HNSW (Hierarchical Navigable Small World) Graphs

  - FAISS (Facebook AI Similarity Search)

Text Preprocessing and Enrichment:

- The data is processed using **tokenization**, l**emmatization**, **entity recognition**, and other NLP techniques to improve the quality of indexing.
- Enrichment techniques such as **synonym expansion** and **knowledge graph integration** help improve search relevance.

Query Processing:

- When a user submits a query, it is also transformed into a semantic vector, and the index is searched for the most **semantically similar** documents using distance metrics like:

  Cosine similarity

  Euclidean distance

  Dot product similarity

Ranking Mechanisms:

- The retrieved documents are ranked based on their **semantic similarity scores** and additional factors like user preferences, click-through rates, or contextual metadata.

Advantages of Semantic Search Indexing:

Improved Relevance:

- Understands the **intent** behind queries, providing more meaningful results.

Handling of Synonyms and Variations:

- Matches concepts even if different words are used.

Multilingual Capabilities:

- Can support multiple languages by leveraging language models.

Better Handling of Natural Language Queries:

- Works well with conversational and voice search inputs.

Applications of Semantic Search Index

E-commerce Search Engines

- Helps customers find relevant products even if they use different terminology.

2. Knowledge Bases & Document Retrieval

- Assists in finding relevant information across large corpora of documents.

3. Customer Support Chatbots

- Enables chatbots to provide better responses based on intent understanding.

4. Healthcare & Legal Research

- Finds related cases or medical documents with similar underlying meanings.

5. Recommendation Systems

- Enhances personalized recommendations based on user interests.

Popular Tools and Technologies Used in Semantic Search Indexing

Vector Search Databases:

- FAISS (by Facebook)
- Annoy (by Spotify)
- Milvus
- Elasticsearch with dense vector support

Pretrained NLP Models for Indexing:

- BERT
- GPT-based models
- Sentence Transformers (SBERT)

Challenges in Building a Semantic Search Index

Computational Cost:

- Generating and searching through vector embeddings requires significant resources.

Data Quality Dependence:

- Requires clean, well-structured text data for effective indexing.

Explainability Issues:

- Hard to interpret why certain results were retrieved compared to keyword-based searches.

Python Code

```python
import faiss
import numpy as np
from sentence_transformers import SentenceTransformer

# Step 1: Load the pre-trained embedding model
model = SentenceTransformer('all-MiniLM-L6-v2')  # A lightweight and efficient
model
# Sample text corpus
documents = [
    "Artificial Intelligence is transforming the world.",
    "Machine learning is a subset of AI.",
    "Natural language processing enables machines to understand text.",
    "Deep learning is driving advances in AI research.",
    "AI applications are widespread across various industries."
]

# Step 2: Convert text data into embeddings
embeddings = model.encode(documents)

# Step 3: Create a FAISS index for fast similarity search
dimension = embeddings.shape[1]  # Get the dimension of the embeddings
index = faiss.IndexFlatL2(dimension)  # L2 (Euclidean distance) index

# Add embeddings to the index
index.add(np.array(embeddings, dtype=np.float32))

print(f"Number of documents indexed: {index.ntotal}")

# Step 4: Perform a search query
query = "What is machine learning?"
query_embedding = model.encode([query])

# Search the index
k = 2  # Number of top results to retrieve
distances, indices = index.search(np.array(query_embedding, dtype=np.float32), k)

# Display the results
print("\nTop matches:")
for idx in indices[0]:
    print(f"- {documents[idx]}")
```