Semantic Search

Meaning-based Information Retrieval in NLP

1. Introduction

Semantic Search retrieves information based on *meaning and context*, not just keywords. It leverages NLP and vector embeddings to match the user's intent with the most relevant information.

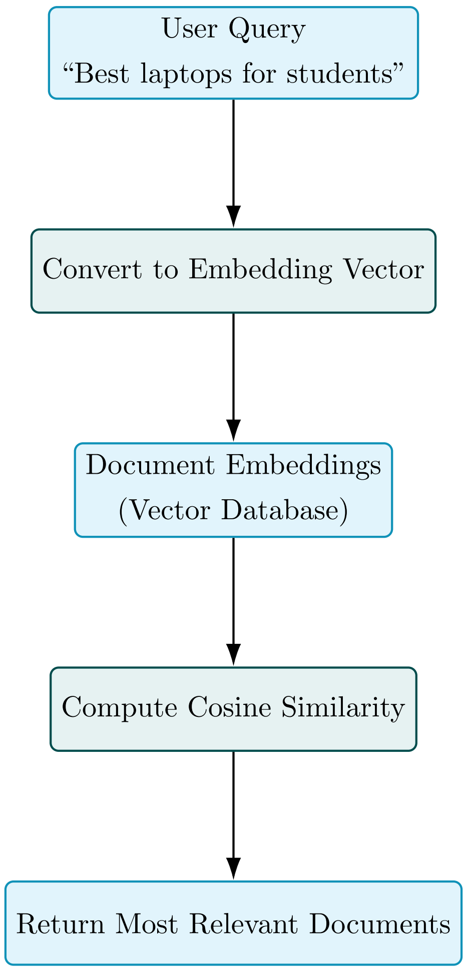
Unlike traditional keyword search, semantic search interprets the *concepts and relationships* in queries and documents — enabling more intelligent retrieval.

2. Why Semantic Search is Needed

Traditional keyword search cannot understand synonyms, paraphrases, or context. For example, searching "affordable notebook for students" may miss documents with "cheap laptop for learners". Semantic search solves this by comparing meanings using vector embeddings and similarity metrics.

3. How Semantic Search Works

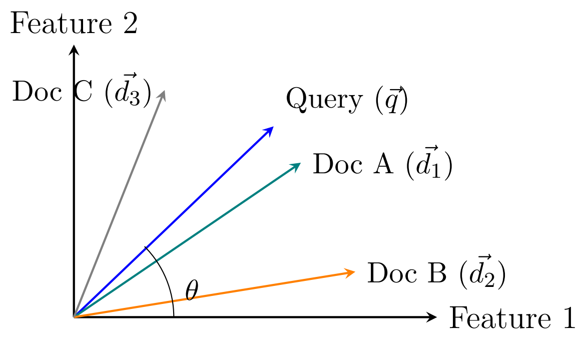
1. Convert all documents and queries into vector embeddings (e.g., Sentence-BERT, OpenAI Embeddings).
2. Store document vectors in a **vector database** (FAISS, Pinecone, Chroma).
3. Convert the user query into a vector.
4. Compute **cosine similarity** between the query vector and document vectors.
5. Return top-k most similar results.


User Query
“Best laptops for students”
Convert to Embedding Vector
Document Embeddings
(Vector Database)
Compute Cosine Similarity
Return Most Relevant Documents

4. Keyword Search vs Semantic Search

Feature	**Keyword Search**	**Semantic Search**
Matching Type	Exact word match	Concept/meaning match
Synonym Understanding	No	Yes
Query Flexibility	Low	High
Example	“buy cheap laptop”	“affordable notebook for students” $\rightarrow$ matches
Technology	TF–IDF, Inverted Index	Embeddings + Vector Similarity

5. Vector Space Visualization



Smaller $\theta \Rightarrow$ higher cosine similarity $\Rightarrow$ more relevant document.

6. Real-World Example: AI Document Q&A System

Query: "How to learn machine learning quickly?"
Document A: "Fastest way to master ML algorithms." **Document B:** "Top programming languages for data science."

Convert each into embeddings:

$$\vec{q} = [0.12, -0.45, 0.89, \dots], \quad \vec{A} = [0.11, -0.47, 0.91, \dots]$$

Compute cosine similarity:

$$\text{sim}(\vec{q}, \vec{A}) = \frac{\vec{q} \cdot \vec{A}}{||\vec{q}||\, ||\vec{A}||}$$

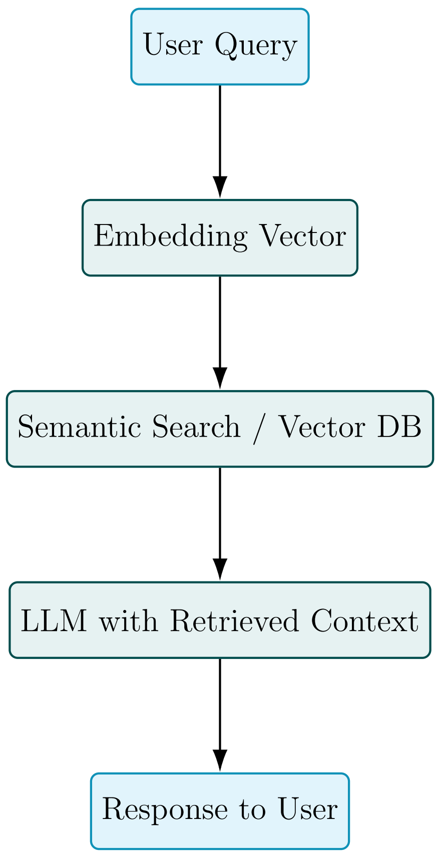
High similarity $\rightarrow$ Document A is retrieved as the most relevant answer.

7. Semantic Search in RAG Systems

RAG (Retrieval-Augmented Generation) uses semantic search to provide LLMs with context-aware knowledge:

1. User query is embedded into a vector.
2. Semantic search retrieves the top relevant documents.

3. Retrieved content is injected into the LLM prompt.
4. LLM generates a response grounded in retrieved knowledge.



8. Python-style Implementation Example

```
from langchain.embeddings import OpenAIEmbeddings
from langchain.vectorstores import FAISS

docs = ["AI is transforming business.",
        "How to learn ML fast?"]

embeddings = OpenAIEmbeddings()
vector_store = FAISS.from_texts(docs, embedding=embeddings)

query = "best way to study machine learning"
results = vector_store.similarity_search(query, k=1)
```

```
print(results[0].page_content)
```

9. Applications

- RAG-based LLM assistants.
- AI-powered chatbots and knowledge retrieval systems.
- Document search engines (legal, academic, corporate).
- E-commerce product recommendation.
- FAQ and customer support automation.

Summary

Semantic search enables *meaning-based information retrieval*, bridging the gap between human intent and raw text. It underpins RAG systems, AI assistants, and modern NLP applications — making information access smarter, context-aware, and highly relevant.