Assignment Requirements:

1. In your own words, explain:

Q-1: What is a vector embedding?
Ans. Vector embedding are numerical representations of text, image, and other data types in multi-dimensional arrays. It captures semantic meaning of the input and groups the same context near-by in the vector graph.

Q-2: What problem does a vector database solve?
Ans. Unlike traditional database, vector database can store any type of complex data in high dimensional space and help LLMs retrieve data in efficient way. Instead of matching “exact” values, it finds vectors having “closest meaning” using similarity search feature.

Q-3: Why does RAG require a vector store?
Ans. RAG systems process data through custom documents that are of unstructured and inconsistent type, they need to fetch relevant answers which are similar in meaning against the search query. These operations are only possible via vector store that have embeddings of the documents and with multiple techniques like HNSW indexing, it can fetch accurate results in a short period of time on a dataset that is not pre-trained with the model.

2. Define the following clearly:

- Cosine similarity: Cosine similarity is a widely used similarity metric that determines how similar two data points are based on the direction they point rather than their length or size. It is especially effective in high-dimensional spaces where traditional distance-based metrics can struggle.

- Chunking: Chunking is the process of splitting large amount of text into small groups of characters by defining the allowed length of characters to fit into that group, called chunk size. Chunking prevents overloading the context window of a LLM which helps reduce hallucinations, thus providing better results.

- Metadata filtering: The process of providing extra information such as source, tags, page-number, chapter, etc. that helps LLM filter out unnecessary data and only focus on vectors that points to the given information for faster similarity search and less memory occupancy.

- HNSW indexing: The HNSW index is a graph-based indexing algorithm that can improve performance when searching for high-dimensional floating vectors. It offers excellent search accuracy and low latency, while it requires high memory overhead to maintain its hierarchical graph structure.

3. Scenario Question:

A support chatbot needs to answer questions from a knowledge base containing 200 PDFs.
Explain step-by-step how you would:

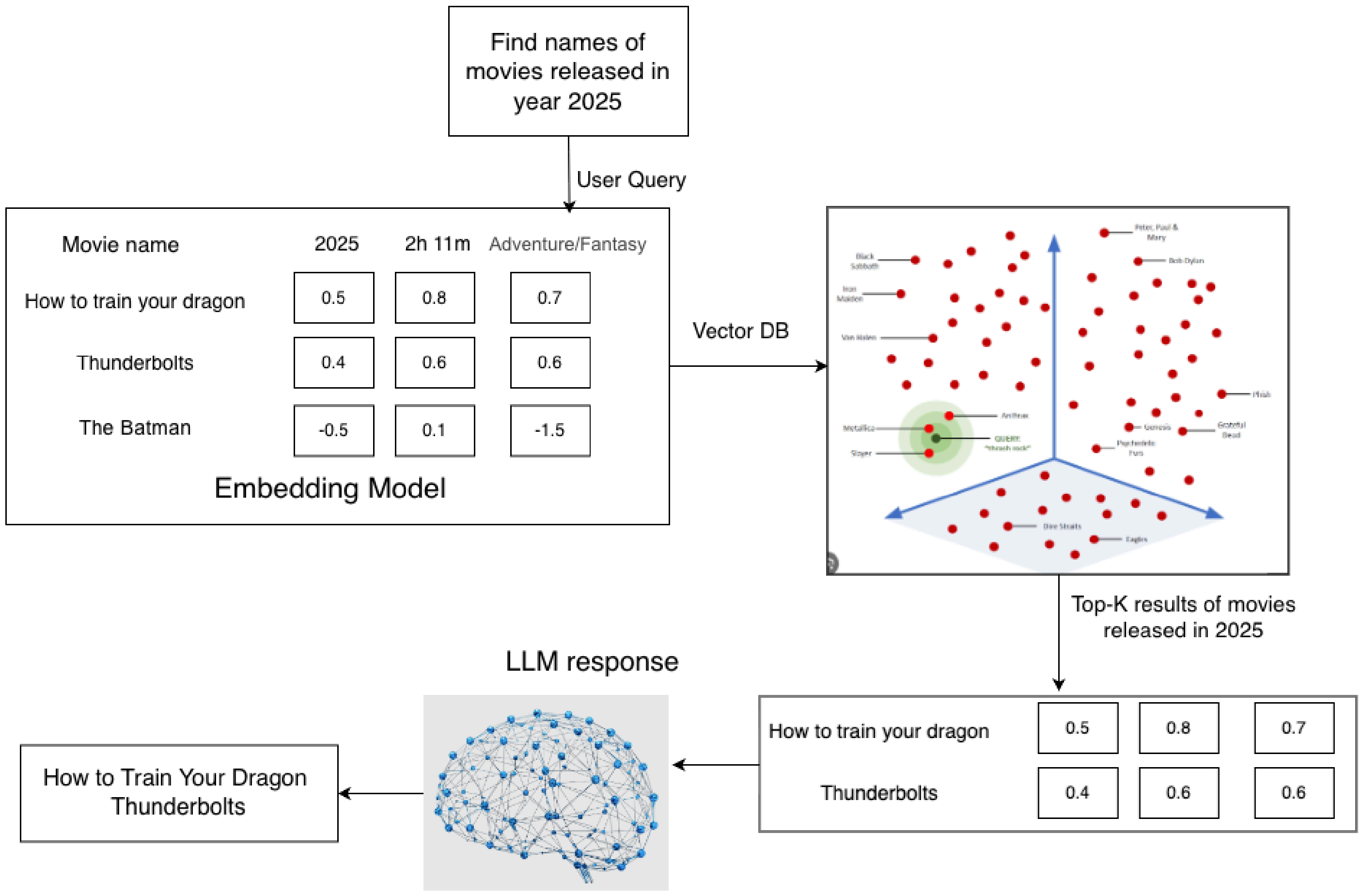
1. Prepare documents
    - Using PyPDF module, feed the model with the text contents of support documentation.
    - Use Recursive-Character-Text-Splitter to create chunks of text from the document.
2. Generate embeddings
    - Select an embedding model (For eg: Hugging face’s all-MiniLM-L2-v6)
    - Encode the document to the model using sentence transformation.

3. Store them
    - Create an embedding object and store the encoded document into it.

4. Retrieve relevant answers using similarity search
    - Choose a similarity metrics from cosine, Euclidean/Manhattan distance.
    - The most recommended similarity metrics is Cosine similarity. We will use that here.
    - From scikit-learn library, import cosine_similarity module, parse the embeddings into the cosine_similarity and print the score.

4. Conceptual Architecture:

Draw or describe a high-level flow for
"User query → Embedding → Vector DB → Top K results → LLM response".



Movie name	2025	2h 11m	Adventure/Fantasy
How to train your dragon	0.5	0.8	0.7
Thunderbolts	0.4	0.6	0.6
The Batman	-0.5	0.1	-1.5

How to train your dragon	0.5	0.8	0.7
Thunderbolts	0.4	0.6	0.6

5. Optimization Question:

Explain three ways to improve retrieval quality in a RAG system using vector databases.

Answer:

A. **Implementing a Two-Stage Retrieval (Top-k + Re-ranking)**: While standard RAG uses top-k retrieval based on vector similarity, the initial search (using Bi-Encoders) can sometimes prioritize keyword overlap over deep semantic relevance.
B. **Utilizing HNSW for Optimized Search Space:** Hierarchical Navigable Small Worlds (HNSW) is an indexing algorithm that creates a multi-layered graph structure for proximity searching.
C. **Advanced Metadata Filtering & Hybrid Search:** Using Metadata Filtering allows you to combine structured data (dates, categories, authors) with unstructured vector embeddings.