**Title: Step-by-Step Approach to Building a Semantic Video Chunk Graph**

**End-to-End Pipeline Overview**

Before building the semantic graph, here’s how we transform raw video data into meaningful graph nodes:

1. **Video → Frames**
   * Each video is sampled at a fixed interval (e.g., 1 frame/sec)
   * Result: A set of image frames per video
2. **Frames → Chunks**
   * Consecutive frames are grouped into a chunk (e.g., 10-frame window)
   * Each chunk represents a meaningful portion of a video (scene/subscene)
3. **Chunks → Metadata Enrichment**
   * Captions generated using image captioning model (e.g., ViT-GPT2 or BLIP)
   * Object detection via YOLO
   * Named entities and noun chunks extracted from captions
   * Each chunk is now enriched with:
     + Captions
     + Caption Entities (via spaCy)
     + Objects (via YOLO)
     + Combined all\_entities
4. **Chunks → Graph Nodes**
   * Each chunk becomes a node in a graph
   * Connections (edges) represent semantic similarity between chunks

**Step-by-Step Plan:**

**1. Load Enriched Chunk Metadata**

* Read the enriched pickle file containing per-chunk:
  + chunk\_id, captions, caption\_entities, objects, all\_entities
  + frame\_paths (optional for visual embeddings)

**2. Initialize Graph and Embedding Model**

* Create a blank graph G using networkx.Graph()
* Load a caption embedding model: SentenceTransformer('all-MiniLM-L6-v2')

**3. Add Nodes to the Graph**

* Each node is a chunk: G.add\_node(chunk\_id, \*\*chunk)
* Store all metadata inside the node for easy retrieval later

**4. Embed Captions**

* For each chunk, concatenate all captions into one text block
* Generate the caption embedding using the SentenceTransformer
* Normalize the embedding vector

**5. Compute Pairwise Similarity Between Chunks**

* Loop over all pairs of chunks (i, j)
  + Compute **caption embedding similarity** using cosine similarity
  + Compute **entity overlap** using Jaccard similarity: |A ∩ B| / |A ∪ B|
  + Combine the scores with weighted sum: weight = alpha \* caption\_sim + beta \* entity\_overlap

**6. Create Edges**

* If weight > threshold (e.g., 0.3), create an edge: G.add\_edge(chunk\_i, chunk\_j, weight=weight)

**Tunable Parameters**:

* alpha, beta: Weighting between embedding similarity and entity overlap
* threshold: Edge inclusion cutoff
* num\_sample\_frames: Number of frames used per chunk for captioning