MPDW Project Report — Phase 2

David Castro[1,2], Yaroslav Hayduk[1,3], and Bruno Baptista[1,4]

[1] FCT UNL, Department of Computer Science
[2] 60973, djc.castro@campus.fct.unl.pt
[3] 60739, y.hayduk@campus.fct.unl.pt
[4] 59815, bm.baptista@campus.fct.unl.pt

1 Introduction

This project addresses the challenge of semantic video moment retrieval using transformer-based architectures. The goal is to create a system that can understand user queries and retrieve temporally relevant segments ("video moments") from long videos.

In phase 1, we focused on understanding and building embedding spaces, indexing video captions, and enabling semantic search using dual encoders and OpenSearch. This involved parsing the ActivityNet Captions dataset (used as our base for video content and metadata), selecting key videos and processing their moments, extracting representative keyframes, computing both textual and visual embeddings, and indexing and querying the data using OpenSearch with support for k-nearest-neighbor search.

In Phase 2, we extended this foundation to support cross-modal retrieval and multimodal reasoning. By embedding both captions and keyframes into a shared representation space using CLIP, we enabled text-to-image, image-to-text, and same-modality semantic retrieval. Additionally, we integrated LLaVA, a large vision-language model, to support visual question answering (VQA). In this setup, the system retrieves a relevant video frame for a given visual question and uses LLaVA to generate a natural language response based on the content of that frame.

These additions allow the system to move beyond simple retrieval and support more complex, open-ended interaction with video content through both cross-modal search and generative question answering.

To gain deeper insight into model behavior, we also explored CLIP interpretability by visualizing similarity scores and attention heatmaps between captions and video frames.

2 System Architecture

Our Phase 2 system forms a five-stage pipeline that converts raw ActivityNet videos into an indexable, query-able, and interpretable LVLM service.

1. **Pre-processing**
   - Trim videos to annotated moments and extract key-frames.
   - Merge timestamped captions from `val_1.json` and `val_2.json`.
2. **Multimodal Embedding Generation**
   - *Sentence-BERT MPNet* → 768-d `caption_vec` (semantic text search).
   - *CLIP ViT-B/32 – text encoder* → 512-d `visual_caption_vec`.
   - *CLIP ViT-B/32 – image encoder* →512-d `keyframe_vec`.
3. **Indexing (OpenSearch)** Three `knn_vector` fields, HNSW (`M`=16, `ef`=256), plus metadata (timestamps, resolution).
4. **Cross-modal Retrieval API** `search(query, modality)` dispatches to one of *text→image*, *image→text*, *image→image* and returns top-k ranked hits.
5. **Retrieval-Augmented VQA** The best key-frame is fed, together with the user question, to *LLaVA-Phi-3*. Answers and CLIP similarities are cached and logged.
6. **Interpretability Module** Generates (i) temporal similarity curves, (ii) Grad-CAM heat-maps for 10 frames, and (iii) a caption × frame contrastive matrix.

2.1 Embedding Representations and Index Schema

We moved from a *dual* to a **tri-encoder** design so that the text and image branches share the same CLIP space. Three independent vector fields are written to OpenSearch (Table 2.1).

- **Sentence-BERT (MPNet)** Captions → 768-d `caption_vec`. Used for classic semantic *text→text* retrieval baselines.
- **CLIP ViT-B/32 — text encoder** Captions → 512-d `visual_caption_vec`. Aligns captions with images and enables *text→image* queries.
- **CLIP ViT-B/32 — image encoder** Key-frames → 512-d `keyframe_vec`. Supports *image→image* and *image→text*.

Field	Model	Dim.
`caption_vec`	SBERT MPNet	768
`visual_caption_vec`	CLIP (text)	512
`keyframe_vec`	CLIP (image)	512

Document granularity. One index document corresponds to a *(key-frame, caption)* pair. A key-frame that overlaps two temporal moments is therefore duplicated, each time with the relevant caption and its own ID. Metadata stored alongside the vectors now includes: `video_id`, `start_timestamp`, `end_timestamp`, `duration`, `resolution`, and the local `keyframe_path`. All vectors are ℓ_2-normalised before insertion so cosine similarity reduces to the inner-product HNSW metric configured in OpenSearch.

Contextual embeddings: For visualizing contextual embeddings, we utilized the caption of one of our videos: "A man is in a skateboard track, then he throws a bowling ball that goes around and hits the pins". The resulting graphs of the layers zero to eleven can be seen in Fig. 1.

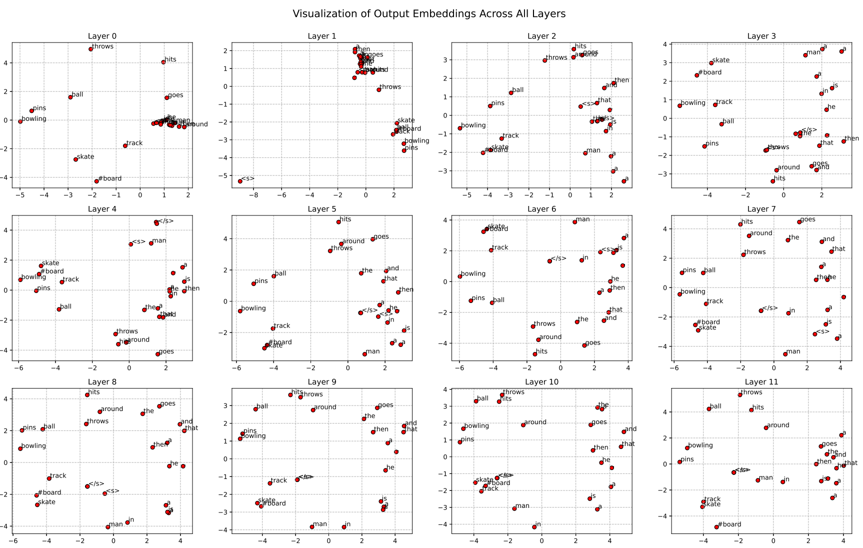

Fig. 1. Contextual embeddings

Surprisingly, in the first layer, the embeddings are very similar to their values in the eleventh layer, but this outcome is merely coincidental, while in the second layer we get something that would be more expected, that is, most embeddings are fairly close together with seemingly no relation or logic. With the progression of the layers, we can see the embeddings disperse and form sensible groups. Namely, we can see the terms "bowling" and "pins" very close together and far from most other words. This makes sense given how specific and relatively rare those words are. The same can be observed for skate, track and board — the suffix of skateboard, hence the proximity. Lastly, the most obvious cluster, on the right: the most common english words. Words such as "a", "the", "that" and "and" which, due to the frequent and indiscriminate use that is given to them, hold very little semantic value and weight in the meaning of sentences.

Positional embeddings: By repeating the previous exercise with a sentence of the same word repeated 20 times instead, what we get is the positional embeddings of our sentence (see Fig. 2). The result is a phenomenon that can be explained by the gradual decrease of positional importance throughout the sentence. Stated in simpler terms, the farther a token is from the first word, the smaller the importance of its position. The first word is the one we can see down at the bottom, the farthest from all the others, as its position has the highest relevance. As we move forth in the sentence, the position relevance gradually decreases and we get increasingly similar embeddings. Due to this, we see the converging effect that results in the cluster of points we see at the top, with all the embeddings of the last words close together.

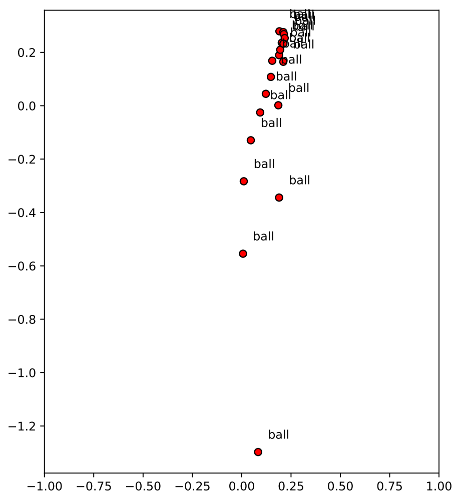

Fig. 2. Positional embeddings

2.2 Large Vision and Language Models

While Phase 2 delves deeper into large vision and language models, in Phase 1 we already used pretrained transformers:

– **MPNet (All-MPNet-Base-V2)** for generating context-aware text embeddings.
– **CLIP (ViT-B/32)** to embed keyframes into a multimodal representation space.

In Phase 2, we extended the use of CLIP to achieve full **cross-modal retrieval** functionality by embedding both keyframes and captions using CLIP's respective image and text encoders. This allows us to represent both modalities in the same embedding space, enabling:

– **Text-to-image retrieval**, where a natural language query retrieves relevant keyframes.
– **Image-to-text retrieval**, where a visual frame can retrieve semantically aligned captions.
– **Image-to-image** and **text-to-text retrieval**, leveraging the shared embedding space for similarity search across same-modality inputs.

All CLIP embeddings, whether from keyframes or captions, were indexed in OpenSearch, allowing for real-time similarity search.

We also integrated **LLaVA (Large Language and Vision Assistant)** for visual question answering (VQA). LLaVA combines CLIP's visual encoder with a language model fine-tuned for multimodal reasoning.

In our system, we used retrieval-augmented VQA:

1. A visual question is submitted.
2. The embedding for the question is computed using CLIP's text encoder.
3. A k-NN search is performed to retrieve the most relevant keyframe based on semantic similarity in the shared embedding space.
4. Both the retrieved frame and the question are passed to LLaVA.
5. LLaVA returns a grounded, natural language answer to the question based on the content of the frame.

2.3 Attention

Self-attention for Cross-Encoder: To analyze the model's self-attention mechanism we generated every head's self-attention for the input "where is the man? A man is in a skateboard track, then he throws a bowling ball that goes around and hits the pins". When looking at the different heads there are multiple patterns we can identify, giving us an intuition of their purpose. A lot of them also simply pay a general, overall attention to the whole sentence. If we look at figure 3, head 2 looks at every token relatively the same, with a special interest in itself, given the distinctive diagonal. This phenomenon can be seen even more clearly in head 7. Head 3 shows us a very clear diagonal pattern, where each token pays higher attention to itself and the around 3 to 6 right next to it, with virtually no attention to all the others; this could be loosely interpreted as: this head solely pays attention to the current token and the few right next to it. Similar effects can be seen in other heads, such as head 6, 9 and 11. Head 6 seems to be paying most of its attention to the couple of tokens right before and right after the current token. Head 9 attends pretty much only to the two tokens right behind the current token. While finally, head 11, pays attention almost exclusively to the token immediately after.

Self-Attention for Dual-Encoder: In the dual-encoder architecture, the query and document are processed independently, which results in separate self-attention mechanisms for each. To analyze this, we visualized the attention heads for both the query sequence *"Where is the man?"* and the document *"A man is in a skateboard track, then he throws a bowling ball that goes around and hits the pins"*, as shown in Figures 4 and 5.

For the query, attention patterns are more localized and interpretable due to its short length. Many heads show weak diagonals, indicating that each token attends mostly to those ahead of it. In contrast, the document's self-attention shows more diverse behavior. Several heads focus attention across broader spans of the input, capturing syntactic structures or semantic relations within the sentence. For instance, tokens such as *"skate"* or *"hits"* often receive and distribute attention across related nouns like *"board"* or *"pins"*, suggesting an awareness of action-object relationships. Other heads demonstrate local attention, where each token mainly attends to subsequent neighbors.

Unlike the cross-encoder, the dual-encoder architecture does not allow for query-document token interaction within the attention layers. Instead, it relies on the resulting embeddings of the entire sequences to compute similarity. We can clearly see from our results how similar their embeddings are, which greatly enhances that operation's precision.

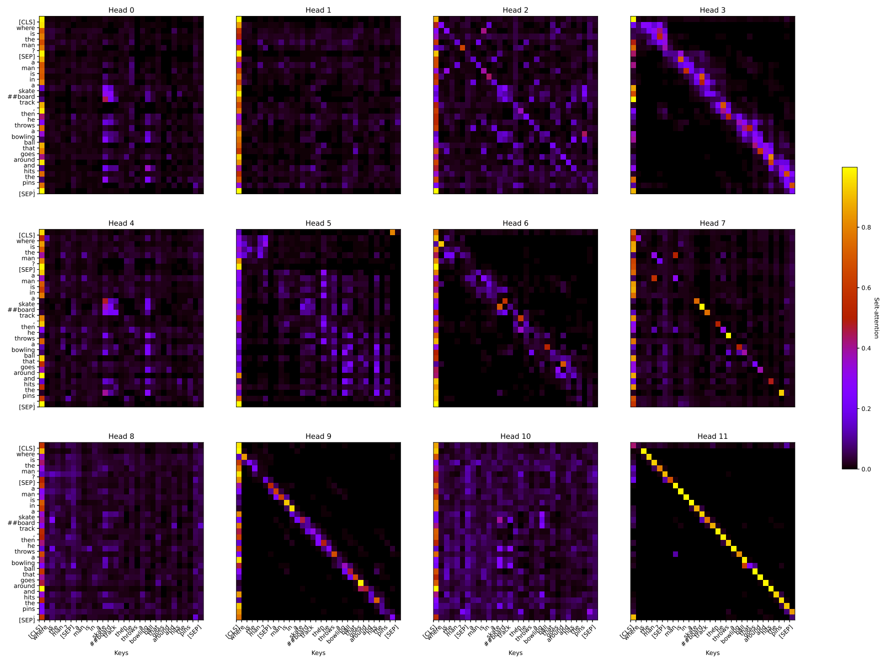

Fig. 3. Self-attention

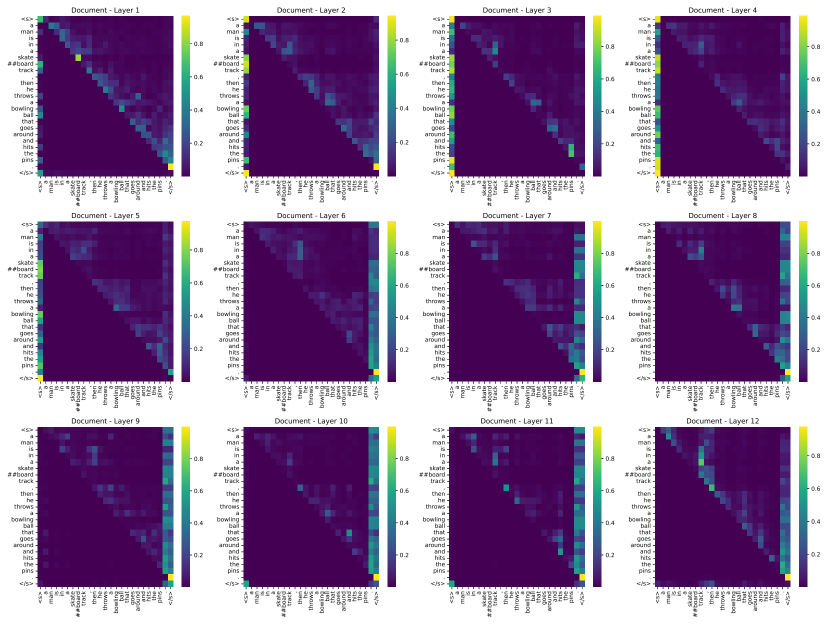

Fig. 4. Self-attention document

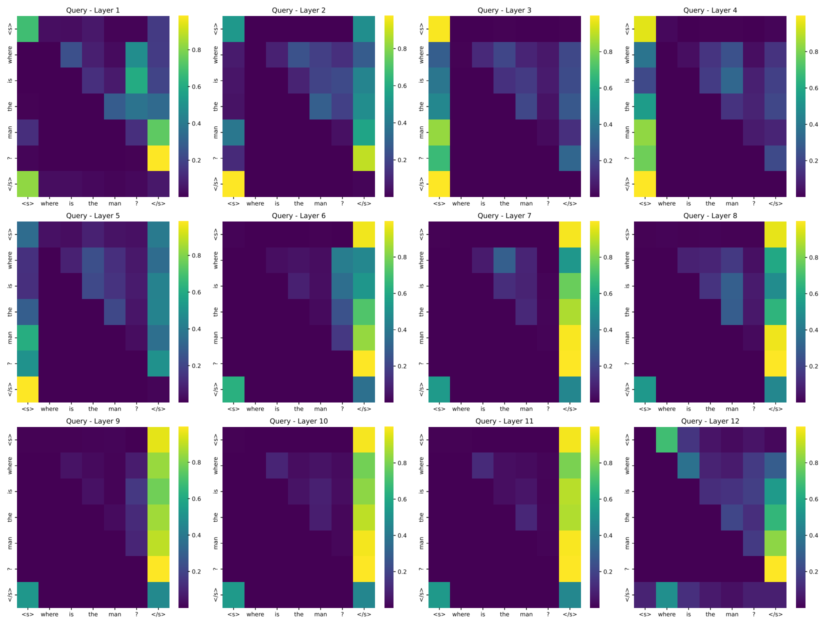

Fig. 5. Self-attention query

3 Evaluation

3.1 Dataset Description

We used the ActivityNet Captions dataset, which provides:

- ∼20k YouTube videos with natural language descriptions of events,
- Timestamp-aligned captions for video moments.

In our implementation, we selected from the dataset 10 videos with a reasonable amount of moments. For each:

- Keyframes were extracted using PyAV by saving either keyframes or one frame per second.
- Corresponding captions were merged and aligned using metadata from three different files: val_1.json, val_2.json, and activity_net.v1-3.min.json.

The resulting data was indexed in OpenSearch using custom mappings to support both text-based and embedding-based (knn) search.