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.

2 Algorithms and Implementation

The system is built on a modular pipeline that prepares video data and metadata for indexing and retrieval using dual encoder architecture. The key components of this implementation include:

- Video moment selection and parsing,
- Frame extraction and keyframe identification,
- Dual encoder embedding generation (text and vision),
- OpenSearch index creation and document ingestion,
- Search using text and embedding vectors (semantic search).

2.1 Embedding Representations

We employed a dual encoder approach using the following pre-trained models:

- **Textual Embeddings:** Generated using the sentence-transformers/all-mpnet-base-v2 model from HuggingFace. Captions were tokenized and passed through the transformer model. We applied mean pooling across token embeddings to obtain a 768-dimensional sentence vector for each video caption.
- **Visual Embeddings:** Generated for extracted keyframes using the ViT-B-32 variant of CLIP, loaded through open_clip. Images were preprocessed and encoded using the CLIP image encoder. The resulting 512-dimensional vectors were normalized.

Each document in our OpenSearch index represents a keyframe extracted from a video. If the timestamp of a keyframe falls within the range of a moment's start and end time, a document is indexed for that (keyframe, moment) pair. As a result, the same keyframe can appear in multiple documents if it belongs to multiple overlapping moments. Each document includes metadata such as the video ID, moment timestamps, caption, keyframe image path, and their respective embeddings computed using MPNet (for text) and CLIP (for image).

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.

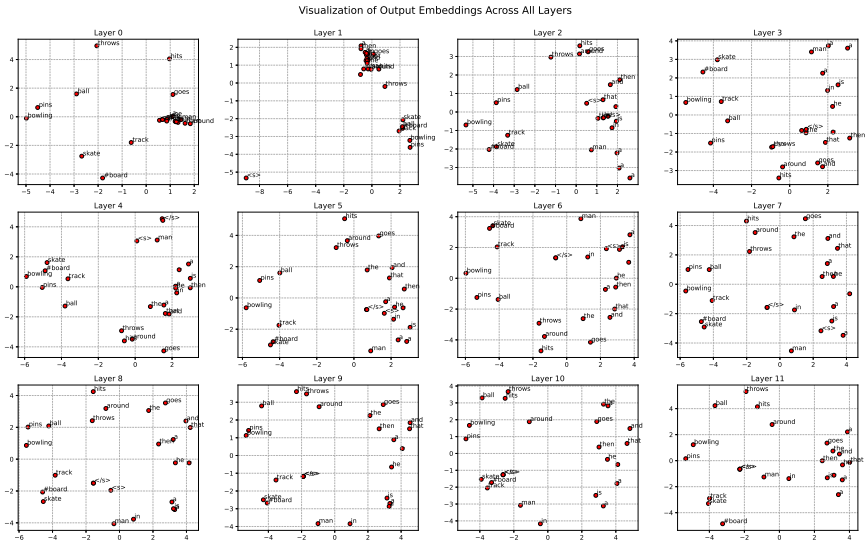

Fig. 1. Contextual embeddings

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.

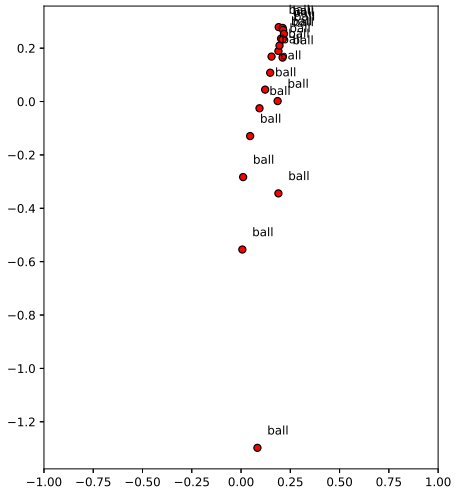

Fig. 2. Positional embeddings

2.2 Large Vision and Language Models

While Phase 2 will delve 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.

These models enable semantic alignment between visual content and natural language descriptions.

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.

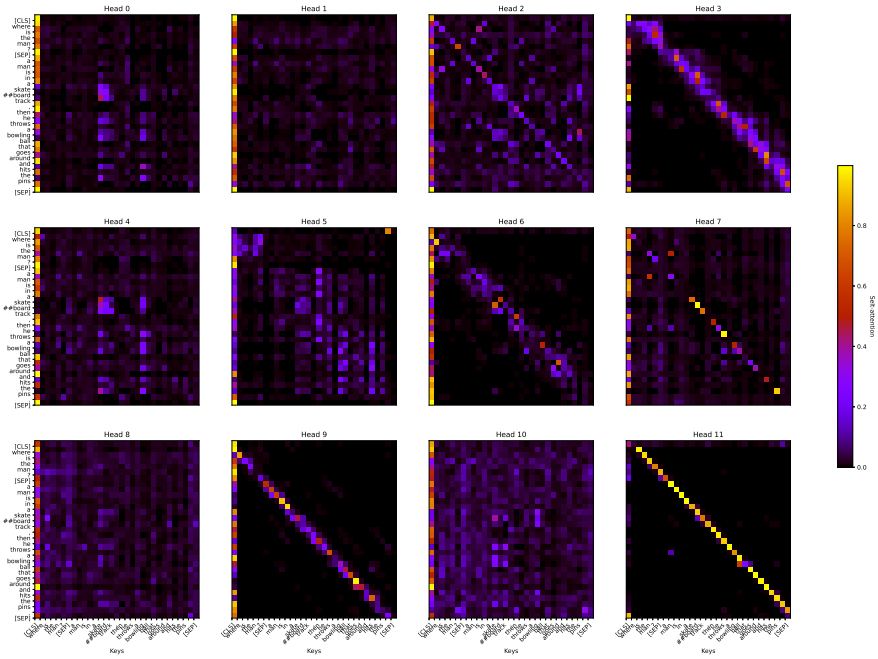

Fig. 3. Self-attention

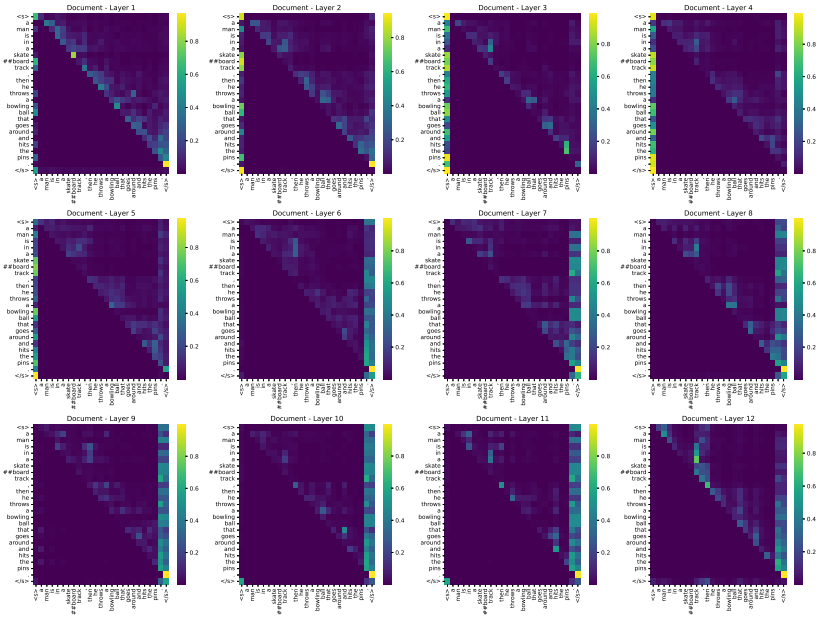

Fig. 4. Self-attention document

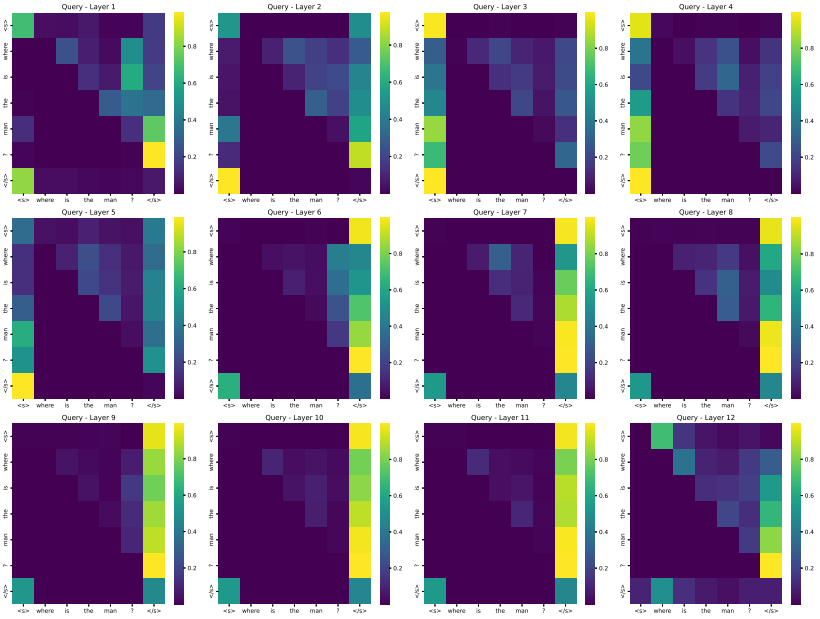

Fig. 5. Self-attention query

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.