Video Dialog

Web Search and Data Mining

Project Guide

João Magalhães
(jmag@fct.unl.pt)

Universidade Nova de Lisboa
2024/2025

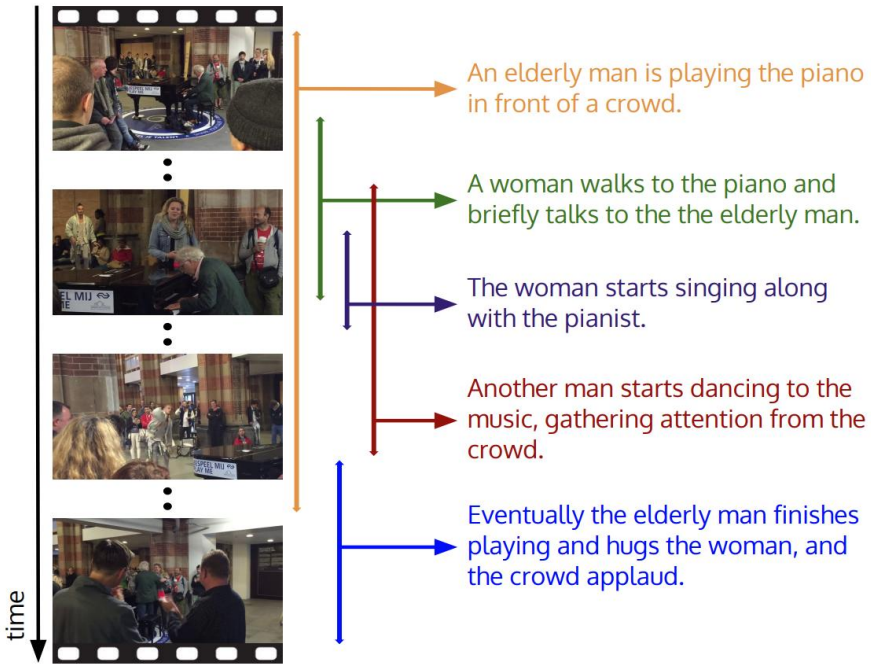
1 Introduction

In this project, students will create a pipeline for moment retrieval in videos using transformer-based architectures. The goal is to enable efficient video moment search with accurate results, enabling applications such as personalized finding the correct moment in a video, do question-answering about video content, and more. By leveraging transformer-based models like CLIP, LLava and Llama 3, you will develop a system that can efficiently retrieve video moments based on user input, allowing for complex cross-modal reasoning and inferences.

1.1 Video Structure

Video content is temporally structured into different levels of granularity. Depending on the task, the video information should be indexed accordingly:

- **Video moments** capture scenes that are semantically coherent. A video moment may contain different video shots.
- **Video shots** are visually coherent frames that were captured with one single camera shot.
- **Video keyframes** are specific frames that are good representations of a temporal range of similar frames.



(Figure from ActivityNet Captions paper)

1.2 Queries and Answers

Your project should support queries with the following elements:

1) natural language questions;
2) natural language questions about selected video moments or key-frames.

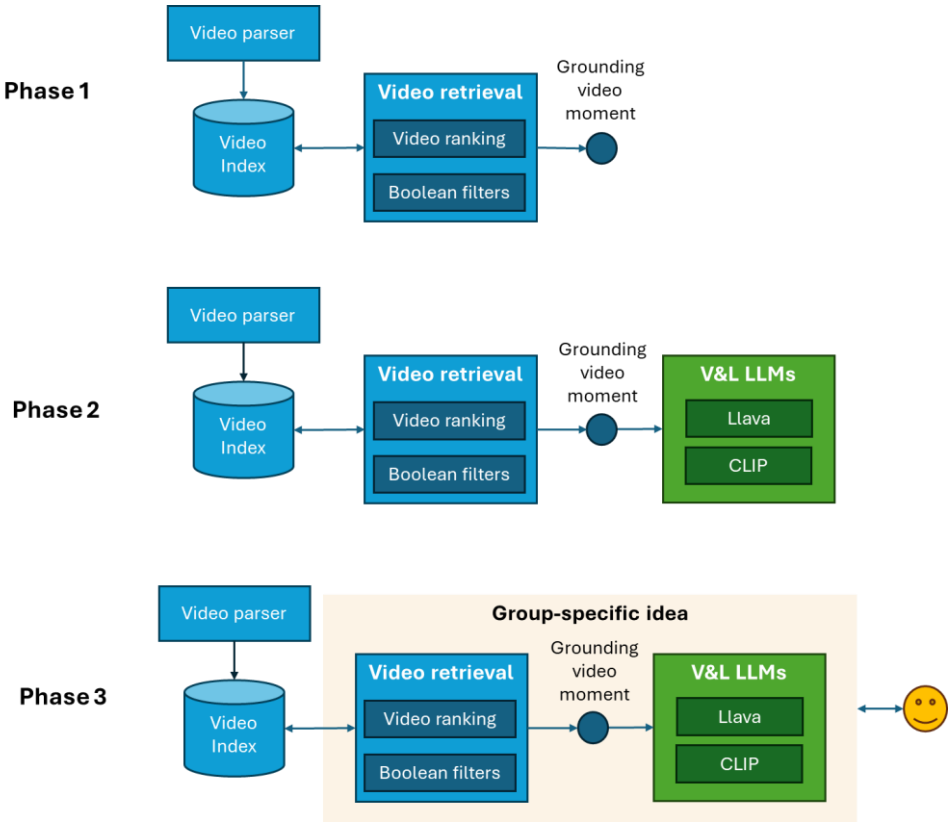
Answers should be computed as:

3) a ranked list of video moments;
4) natural language answers grounded on video moments.

1.3 Project Phases

The project is divided into three parts:

- **Phase 1, Understanding Embedding Spaces:** In this phase, you will create an index of video moments using Dual Encoders and OpenSearch. This will allow your system to retrieve relevant moments based on user input.
- **Phase 2, Large Vision and Language Models:** In the second phase, you will integrate multimodal encoders and decoders to handle cross-modal queries and answer natural language questions.
- **Phase 3, Group Specific Ideas:** In the final phase, you will be able to select one of the proposed topics or to suggest your own topic.



2 Phase 1: Embedding Spaces

Following a natural conversation, your final prototype will be able to engage in a simple conversation to ground the dialogue on a particular video. Like every human, your prototype will have its own KB of the videos and moments.

You should follow the tutorials available on the course Web page and studied during the laboratory classes.

2.1 Video Data and Metadata

In this project you will use the ActivityNet Captions dataset, available at different locations:

- ActivityNet Videos dataset:
  - https://github.com/activitynet/ActivityNet
- ActivityNet Video Captions dataset:
  - https://cs.stanford.edu/people/ranjaykrishna/densevid/
  - https://huggingface.co/datasets/HuggingFaceM4/ActivitiyNet_Captions

2.2 Text-based Search

- Understand how to use the OpenSearch framework. Read the provided code and the documentation.
- Parse the ActivityNet Captions dataset and create the OpenSearch index mappings for the recipe title and description.
- Index the dataset and test the search functionality.
- You can try to optimize the search results.

2.3 Embeddings Neighborhood

The embeddings space provides you with a mechanism to organize data by their semantic similarity. Dual encoders allow you to create semantic embedding representations of documents and queries. Follow the these steps to explore and understand how the embedding space:

- Revise the index mappings to support k-nn vectors.
- Compute the video text embeddings and store them in a file.
- Add the embeddings to your index and test the search functionality (don't forget that you also need to compute the query embeddings in real time).
- Use different queries to search OpenSearch and explore the neighborhood of those queries.

Discuss how the embeddings space organize data and allow for semantic search. Compare it to the text-based search.

2.4 Constrained Embedding Searches

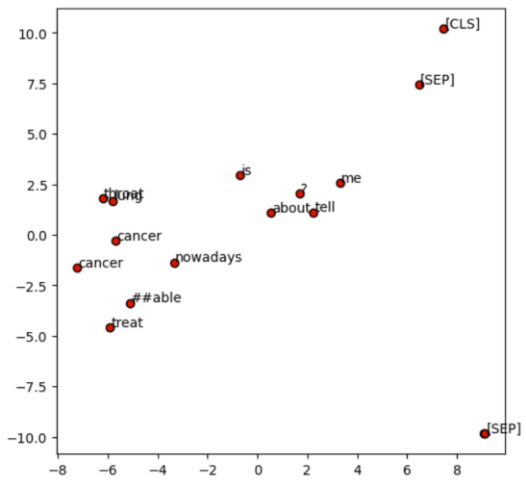
In this phase you should implement different search methods to support searching with natural language questions and search with filters. This is intended for videos with specific characteristics, e.g. actions, entities, length, number of moments, etc. Consider the following possibilities:

- Text based search, e.g., for keyword-based search;
- Embeddings based search; e.g., for semantic search;
- Boolean filters alone, e.g., search by entities;
- Search with boolean filters, e.g. a combination of search methods;

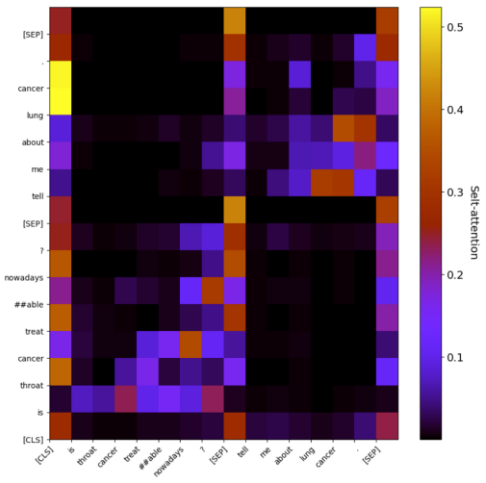
To support the above searches, you should create your own set of mappings to index the video's data. Test different fields and find a strategy to improve results. Using all fields can be sub-optimal and you should propose an optimal set of mappings that support your implementation functionalities.

2.5 Contextual Embeddings and Self-Attention

In this phase of the project you will show your understanding of the self-attention mechanism for vision-language web related tasks. You will also demonstrate your understanding of the Transformer architecture for different tasks.



Word embeddings visualization.



Single-head self-attention visualization.

You should discuss the following points:

1. **Contextual embeddings.** Visualize the contextual word embeddings from layer 0 to layer 11. Observe how they change from layer to layer.

2. **Positional Embeddings.** Consider a text encoder. Insert a sequence of text with the same word repeated 20 times. Visualize the embeddings and the distance across all tokens. Discuss what you observe.

3. **Self-Attention.** Examine the self-attention mechanism of a transformer cross-encoder. Repeat with a dual encoder. Do critical analysis of your observations.

4. **Interpretability.** Compute the attention that each token receives on each layer and visualize. How is this related to the final output of the model?

To address the above questions, you should build on the provided tutorials to visualize word embeddings and self-attention matrices. See the two examples above.

2.6 Tips for Persistent Storage

- Some tasks take a long time to process, e.g. computing the embeddings. Hence, you may wish to put the embeddings in some persistent storage.
- Python pickles allow you to (de)serialize objects: https://docs.python.org/3/library/pickle.html
- HDF5 or Pandas DataFrames provide you with a more structured solution.
- If you wish to store JSON objects, you can do so in the index, however, remember that whenever you delete the index you will lose this data.

2.7 Tips for Index Mappings

When analyzing videos you need to decide what will be your indexing unit (i.e., document in OpenSearch terminology): the entire video, the video moment or a key-frame. For example, your mappings could be done at the level of the video moment, with the following index mappings:

vid_id	YouTube video unique identifier
start_timestamp	video moment start timestamp
end_timestamp	video moment start timestamp
caption_bow	bag-of-words representation of the video movement caption
caption_vec	embedding representation of the video moment caption
video_length	video length
keyframe_vec	embedding representation of the key-frame
…other fields…	

The above mappings assumes that the fine-grain indexing unit is a video moment. You may wish to have finer-grain representation by using keyframes to navigate to specific instants within the video moment.

3 Phase 2: Large Vision and Language Models

In the second phase of the project we will extend the Large Language Encoder Models of the first phase implementation with Vision Encoder models. In addition, we will also study Generative Large Vision and Language Models. Finally, we will leverage interpretability methods to analyze and study how decisions are made in LVLMs.

3.1 Cross-Modal Retrieval

The first objective for this phase of the project is to index video frames images, their captions and support cross-modal retrieval, i.e. retrieve text with image queries, retrieve images with text queries, and retrieve images with image queries.

3.1.1 CLIP encoders

To implement this functionality, you will use a CLIP encoder. CLIP allows you to compute the semantic similarity between a sentence and an image.

CLIP has a dual encoder architecture:

- The image encoder generates an embedding vector for images.
- The text encoder generates an embedding vector for text.
- Both representations exist in the same embedding space and can be used to compute the similarity between text and image data in the visual space, i.e., text is always assumed to describe the contents of the image.

3.1.2 Implementation

To index the videos by their visual contents, you should extract key-frames or one frame for every 2 seconds and compute its CLIP embedding vector. This embedding vector is then stored in OpenSearch for future searches.

Whenever a user submits a query, its text or image must be encoded with the corresponding CLIP encoder (text or image) and the resulting embedding vector should be used as a query for OpenSearch.

You may also explore queries with combinations embeddings, e.g., images+images and text+images.

3.2 Large Vision and Language Models

The second objective for this phase of the project is explore a Large Vision and Language Model, in particular the Llava model. This model will allow your system to answer questions about the content of a visual frame, or to describe its content.

3.2.1 Llava Model

Large Language and Vision Assistant (Llava), is a multimodal AI model designed to understand and generate content based on both visual inputs (images) and textual

instructions. It integrates a vision encoder (such as CLIP ViT-L/14) with a large language model (like Vicuna or LLaMA) to process and respond to multimodal inputs.

LLaVA is trained using a technique called visual instruction tuning, where a language-only model like GPT-4 generates multimodal instruction-following data. This approach enables LLaVA to handle tasks that require understanding both text and images.

3.2.2 Implementation

Use the provided examples to access the Llava API. You may run your own server if you have a GPU with sufficient memory.

Retrieval Augmented VQA: For visual questions, use your CLIP-based index to retrieve candidate frames and select the top retrieved frame. This image frame and the visual question must then be passed to Llava to obtain the answer.

3.3 Interpretability

The introduction of LVLMs and their increased number of parameters, interpreting and explaining model outputs to mitigate hallucination is becoming an important challenge. In light of the need to understand the reasoning behind model responses, there are three different LVLM interpretability approaches:

- **Attention layers** illustrates the interaction among tokens, as determined by the attention between tokens (visual and text).
- **Relevancy maps** illustrates how different components of an input, whether text or image, are relevant to the model's generated output.
- **Causal graphs** explain large vision-language models by learning causal structures over input-output sequences of tokens, i.e. if tokens would have been masked in the input, the model would have generated a different output.

In this part of the project you will explore these three different approaches to LVLM interpretability.

4 Phase 3: Group-specific Ideas

For the third phase you need to implement a part that is unique to your project as a group or individual. This is a list of ideas from which you can choose:

Search/browsing:

1) Improved video search.
2) Video browsing by similarity.
3) Billion scale distributed search.

Conversational agent:

4) Dialog manager with controlled dialog state-tracking.
5) Multimodal Retrieval Augmented Generation.
6) Feedback UI for semi-automatic video annotation.

V&L Explainability:

1. CLIP model explainability.
2. LVLM model explainability.

Literature reviews (min 6 pages, 20 references):

3. Energy and pollution concerns of large vision and language models.
4. Ethical and moral risks of large vision and language models.

Bring your own idea!!! Propose it until May 15.

5 Project Grading, Submissions and Report

The progress accomplished on each phase should be submitted and detailed in a report.

- Phase 1: April 11 20% of the course final grade
- Phase 2: May 12 20% of the course final grade
- Phase 3: June 13 20% of the course final grade

The report should be 15 pages length written during the three phases – annexes are not limited. A suggestion is to write 5 pages on each phase. When you submit the report of each phase, you are allowed to update the text of the previous phase. In exceptional circumstances you may exceed the 15 pages, with high-quality content.

The suggested template is available on Overleaf. The suggested structure is as follows:

1. Introduction
2. Algorithms and Implementation
   - 2.1. Embedding Representations
   - 2.2. Large Vision and Language Models
3. Evaluation
   - 3.1. Dataset description
   - 3.2. Baselines
   - 3.3. Results analysis
4. Critical discussion
5. References

6 Readings

The course site will be updated with readings and tips.

- SentenceBERT: https://arxiv.org/abs/1908.10084
- Facebook MNLI: https://huggingface.co/facebook/bart-large-mnli
- Llama 3: https://arxiv.org/abs/2407.21783
- Llava: https://llava-vl.github.io/
- LVLM-interpret: https://intellabs.github.io/multimodal_cognitive_ai/lvlm_interpret/