Video Dialog

Lina LEKBOURI[1,2[72697]], Marc LICHTNER[1,2[72690]], and Hugo WERCK[1,2[72692]]

[1] NOVA School of Science and Technology (FCT NOVA), https://www.fct.unl.pt/
[2] TELECOM Nancy, Nancy, FRANCE, https://telecomnancy.univ-lorraine.fr/

1 Introduction

The primary objective of this project is to create a system that can understand and respond to natural language queries about video content, enabling users to find precise moments within videos without having to manually search through hours of footage.

Our work will be divided in three parts. In **Phase 1**, we establish the foundation by creating an **index of video moments** using Dual Encoders and OpenSearch.

Phase 2 extends our system by integrating multimodal encoders and decoders, enabling it to handle cross-modal queries and answer natural language questions about video content.

Our objectives of **Phase 3** are not yet decided at this time of the project.

Throughout this report, we document our **methodology, implementation details, challenges encountered, and the results of our experiments**. We also provide a critical analysis of our approach, discussing both its strengths and limitations, and suggesting possible improvements for future work.

2 Algorithms and Implementation

2.1 Embedding Representations

2.1.1 Embeddings neighborhood

The embeddings space provides a powerful mechanism to organize and retrieve data based on **semantic similarity** rather than exact keyword matches. To implement this capability in our video retrieval system, we extended our approach to include nearest neighbors search functionality using dense (knn) vector embeddings.

Revision of OpenSearch index

First, we create a specific field in our mapping to store the vector representations of each video caption (with a dimension of 384, which corresponds to the output dimension of our chosen embedding model):

```
{    "properties": {
        "caption_vec": {
            "type": "knn_vector",
            "dimension": 384,
            "index": True,
            "similarity": "cosine"
        }
    }
}
```

For generating the embeddings, we utilized the `sentence-transformers/all-MiniLM-L6-v2` model (from Hugging Face's Transformers library), designed for sentence embedding tasks.

We then processed all video captions in our dataset, computing their embedding vectors and storing them in a JSON file following the index mapping structure.

After generating all embeddings, we perform semantic searches taking an embedded query and searches the index for documents with embedding vectors closest to the query vector by using cosine similarity, which measures the cosine of the angle between two vectors.

To evaluate the effectiveness of our embedding-based search, we compiled a set of test queries representing different ways users might search for video content and then plot them to see their closeness. As an example:

Query	**Top Result Caption**
"A few other men are shown playing guitars as they sit."	"A few other men are shown playing guitars as they sit."
"A dog running in the park"	"man is doing a javeling throw in a big large field."
"A group of people in a meeting"	"The group plays for the audience, occasionally zooming in on individuals."
"A sunset over the ocean"	"woman used some black painting for make details, put the red stamp on the corner and finished the painting with yellow and reddetails on the flowers."

Table 1: Example queries and their top results using embedding-based search

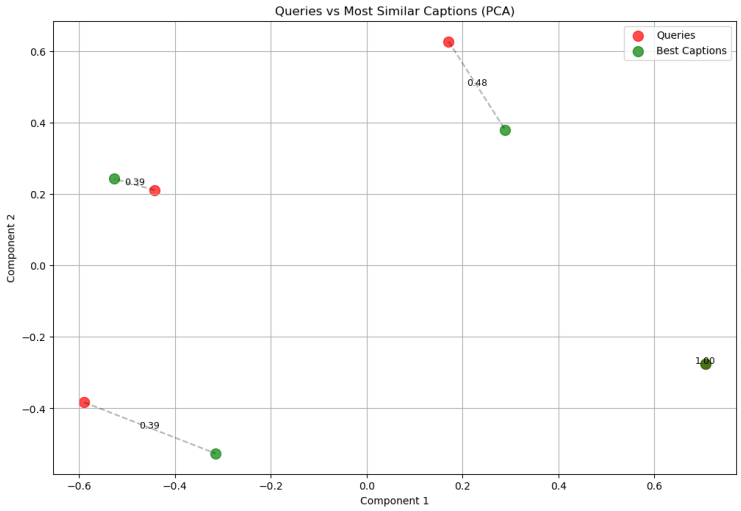

Fig. 1: Plot of Semantic search

The embedding-based search applied here, shows the cosine score between the query, its best match and their position on a 2D plan. We can clearly see that the query and their match are very close to each other, and clearly say that our result are much realistic. Another significant point is putting an exact caption as a query really find the good video, as shown by the point at the bottom right. Though it is costless to use text-base search in this case. In conclusion, by seeing this representation, we can assume that our encoder and our semantic search are doing their job as we wanted.

Indeed, our findings suggest that an optimal approach would combine both methods: using embedding-based search for capturing semantic relationships while retaining text-based search capability for situations requiring exact matching.

2.1.2 Constrained Embedding Searches

To achieve flexible and accurate video moment retrieval, several search strategies were implemented using OpenSearch and semantic embeddings:

- **Text-based search:** Classic keyword matching using the textual field `moment_description` in the index.
- **Embedding-based search:** Semantic retrieval using k-NN over Sentence-Transformer (`all-MiniLM-L6-v2`) embeddings.
- **Boolean filters:** Filtering by structured metadata such as video length.
- **Combined search:** Hybrid queries combining semantic similarity and boolean constraints (e.g., retrieving moments describing a specific action in videos longer than 5 minutes).

The OpenSearch index was custom-designed at the video moment level, including textual captions, embedding vectors, temporal bounds, and metadata. Experiments showed that focusing on a reduced set of well-chosen fields (e.g., caption embeddings and duration) improved retrieval precision and interpretability. Representative query example can be found in 2.1.1.

Empirically, semantic search using embeddings enables retrieval of relevant moments even for paraphrased or descriptive queries, while boolean constraints allow for precise filtering. Compared to pure text search, embedding-based retrieval is more robust to vocabulary mismatch and captures higher-level semantics.

2.1.3 Contextual Embeddings and Self-Attention

To better understand the power of transformer architectures in video retrieval, we analyzed the internal representations of language models and their attention mechanisms, using the `bert-base-uncased` model.

Contextual Embeddings. Here, we used the two following sentences as they have similar words used in slightly different context. Therefore showing the difference between static and contextual embeddings.
Sentence A: "Photosynthesis uses which energy?"
Sentence B:"Oxygen is released by photosynthesis, which uses the energy of sunlight."
At layer 0, the embedding we see is only static because the contextual embedding hasn't started yet. We see that words that appeared in both sentences are close even though they don't exactly mean the same thing as they are not used in the same context. At the last layers, we can see that those words were separated and respectively formed a cluster with their belonging sentence. For instance, the two words energy are extremely close on layer 0 and apart on layer 11. We can even guess that the word energy at the bottom belongs to the second sentence as it formed a cluster with the words "which," "release," "uses," and more. Annex A.1.1.

Positional Embeddings. We analyze a sequence of repeated tokens. Initially, the embeddings are almost identical, as they share the same meaning and lack contextual differentiation. However, as positional information mechanism come into play, the representations gradually diverge. This results in a smooth arc in PCA space, reflecting the incremental integration of context at each position, as shown in Annex A.1.2. Visualization of the pairwise distances between token embeddings is included in Annex A.1.2.

Self-Attention Mechanism. We examined attention matrices generated by the transformer. Visualizations reveal how the model distributes focus among tokens. Comparative analysis between cross-encoder and dual-encoder architectures showed that cross-encoders model token-level interactions, while dual-encoders favor efficiency. As layers and heads progress, attention becomes in-

creasingly focused and semantically precise, evolving from broad structural cues to task-specific understanding. Examples are provided in Annex A.1.3.

Interpretability. After obtaining an overview of the behavior of the attention heads as a function of the layers, we can distinguish three interesting cases in Annex A.1.4 :

- Two colored blocks and two dark blocks: Since the two sentences have been concatenated, each word in the first sentence is followed by those in the second. In this case, each word only pays attention to words in the same sentence, resulting in the presence of two main blocks along the diagonal of the attention matrix: one for each sentence. This symmetry around the diagonal represents the model's ability to recognize each word in its own context, without significant interaction between the two sentences.
- A diagonal shifted by one or more squares: Here, the attention matrix highlights that each word pays particular attention to one or more neighboring words. A shift of one square indicates that attention is primarily focused on the immediately following (or preceding) word, while a shift of several squares shows that attention is focused on more distant words. This reflects the model's ability to capture sequential or long-distance dependencies within the sentence.
- A vertical line: This pattern indicates that many words are paying particular attention to the same word, often the special token [SEP] that separates the two sentences. This means that the model uses this token as a cue or context summary, which can help in understanding the overall relationship between the two sentences after each word has established its own context.

2.2 Large Vision and Language Models

3 Evaluation

3.1 Dataset Description

For this project, the datasets we used are from ActivityNet. We had one file named `activity_net.v1-3.min.json`, which contained data about videos in the following format:

```
{video_id: {duration, subset, resolution, url, annotations: [{segment,
label}]}}
```

For each video in the file, we computed the number of annotations and kept the 10 videos with the highest count. Then, for each of them, we downloaded the video using the corresponding URL. We retained the `video_id`, `duration`, and `url`.

We then used three additional files from ActivityNet to retrieve the timestamps and captions for each moment in the videos. The format was as follows:

`video_id: {duration, timestamps, sentences}`

We were able to retrieve the data for each video using the `video_id`.

All the information for these 10 videos was stored in a variable named `top_videos`, structured as follows:

`{video_id, num_moments, url, duration, moments}`, where `moments` is a list of entries in the format: `{start, end, description, bag_of_words}`.

`start` and `end` represent the timestamps for the moment, and `bag_of_words` is the set of all words from the `description`.

Once we had parsed the JSON files into the `top_videos` structure as described above, we stored the data in OpenSearch. We indexed the data based on the moments, meaning that each entry in OpenSearch corresponds to one moment from a video.

Each document therefore contains the `video_id`, the `start` and `end` timestamps, the `moment_description`, and other information such as the `caption_vector` or `caption_bow`, which will be used later for search.

3.2 Baselines

3.3 Results analysis

4 Critical discussion

5 References

A Annexes

A.1 Phase 1

A.1.1 Contextual Embeddings Visualizations : provides visualizations of contextual embeddings across transformer layers.

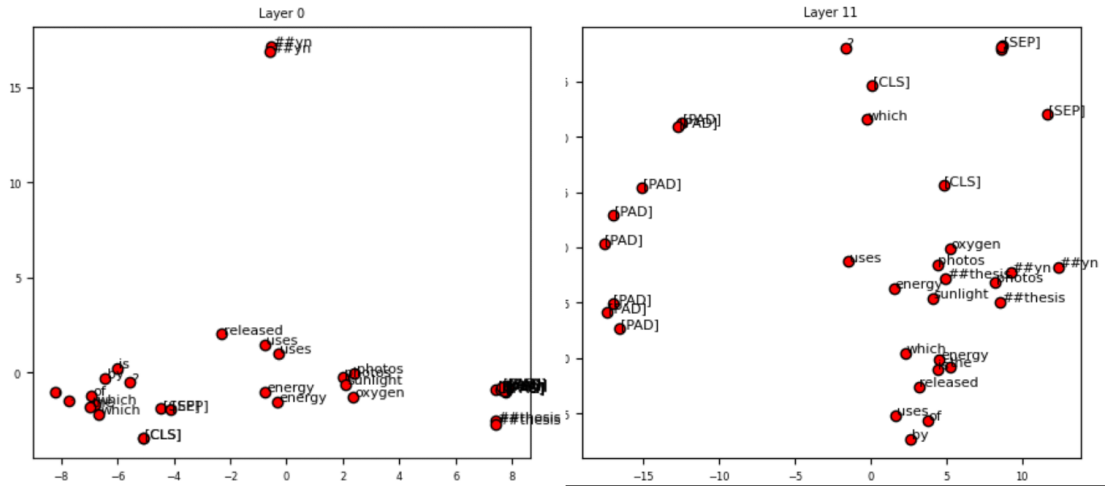

Fig. 2: PCA visualization of token embeddings of layers 0 and 11 (`bert-base-uncased`).

A.1.2 Positional Embeddings : shows the pairwise distances between positional embeddings for repeated tokens, highlighting how positional information is encoded even for identical input tokens.

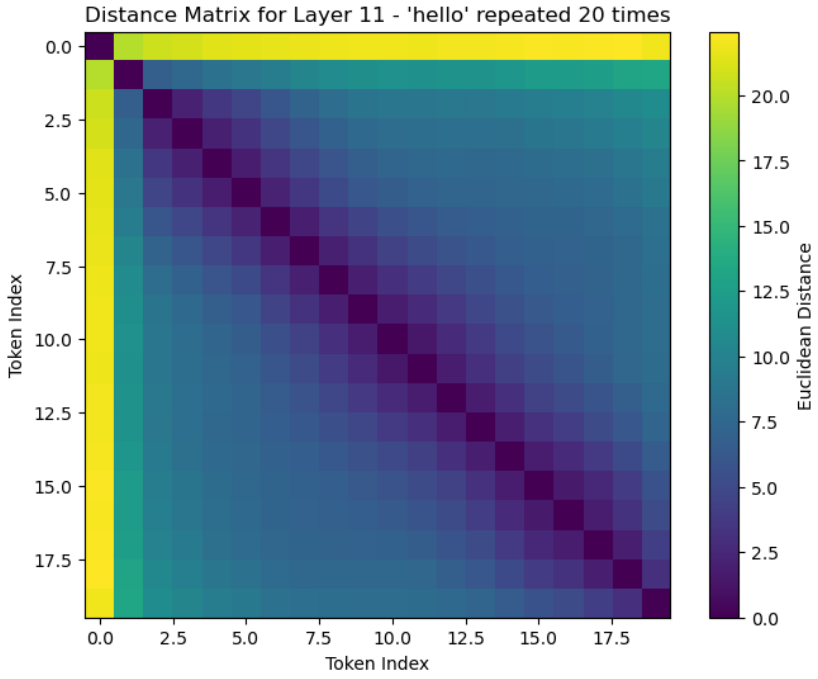

Fig. 3: Pairwise distances between positional embeddings for repeated tokens.

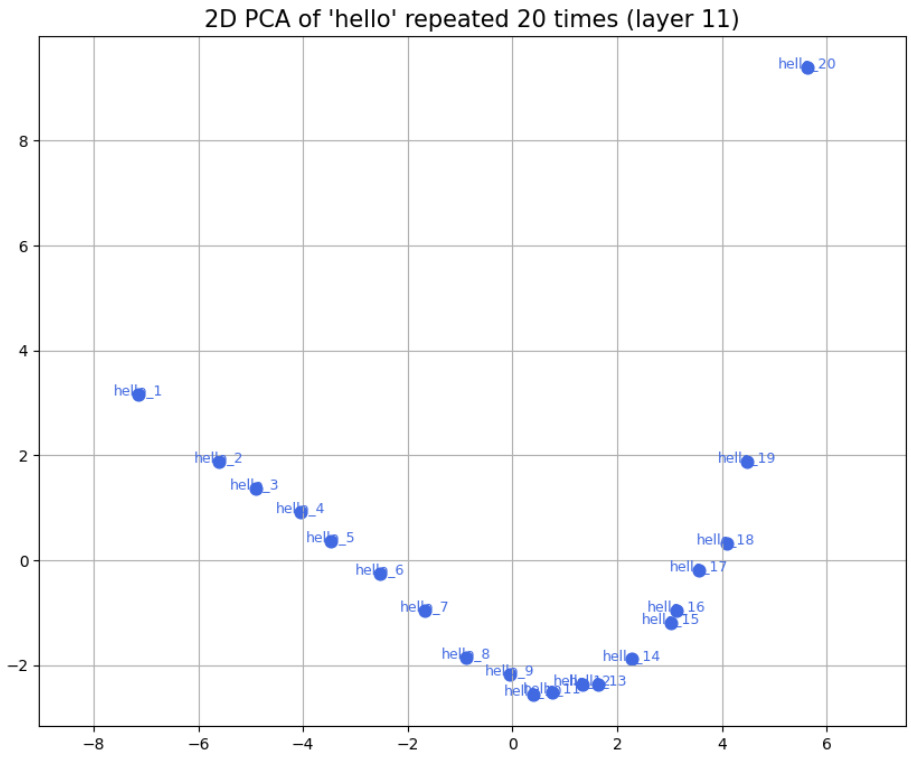

Fig. 4: Embeddings plot for repeated tokens.

A.1.3 Self-Attention Maps : presents self-attention matrices for both cross-encoder and dual-encoder architectures.

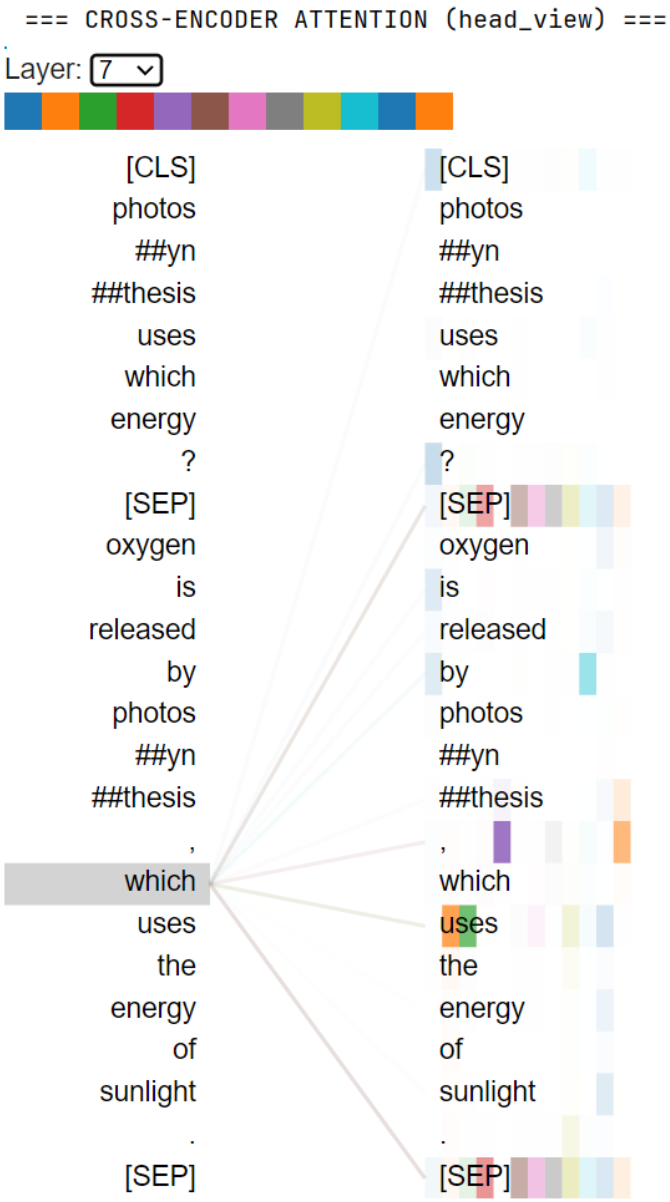

Fig. 5: Self-attention matrix for a cross-encoder, word 'which' at layer 7 (`bert-base-uncased`).

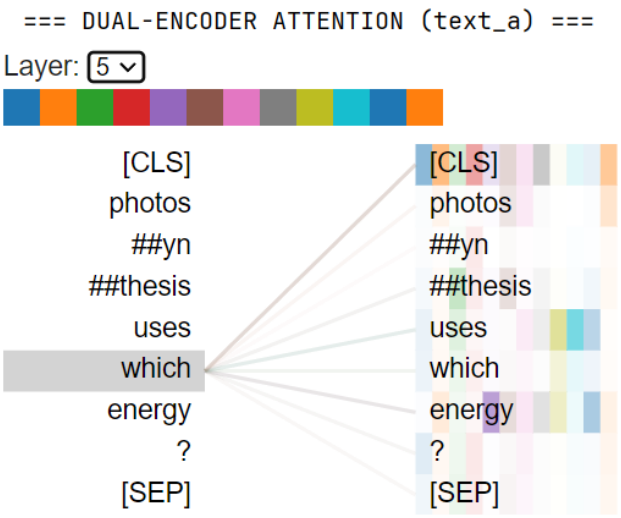

Fig. 6: Self-attention matrix for a dual-encoder, layer 5 from text a (`all-MiniLM-L6-v2`).

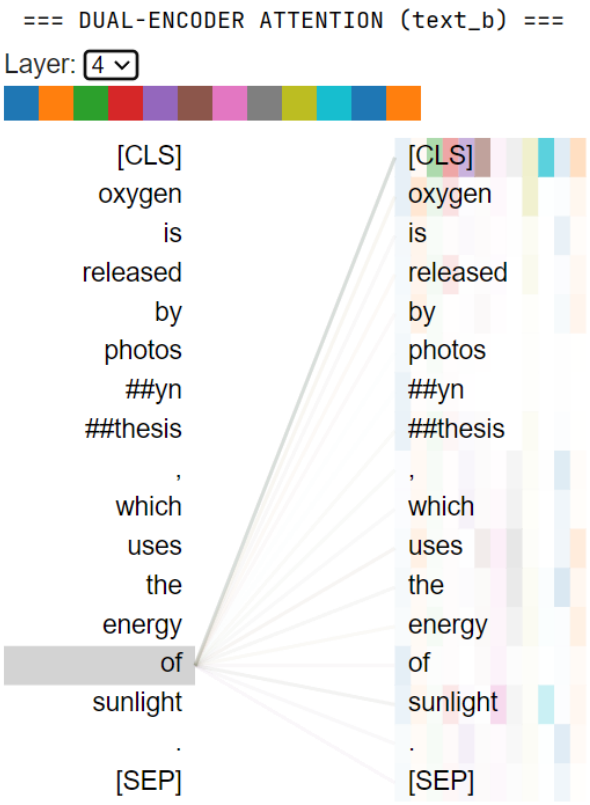

Fig. 7: Self-attention matrix for a dual-encoder, layer 4 from text b (`all-MiniLM-L6-v2`).

A.1.4 Interpretability. :

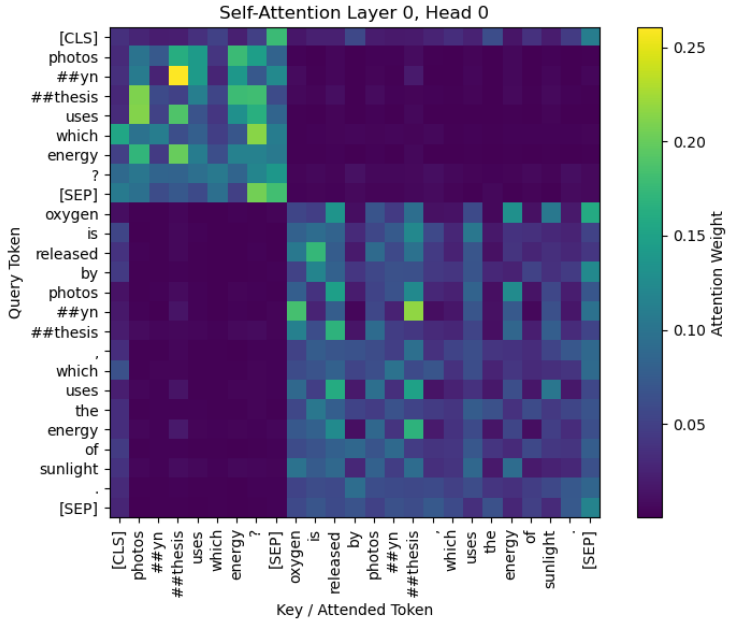

Fig. 8: Layer 0, Head 0

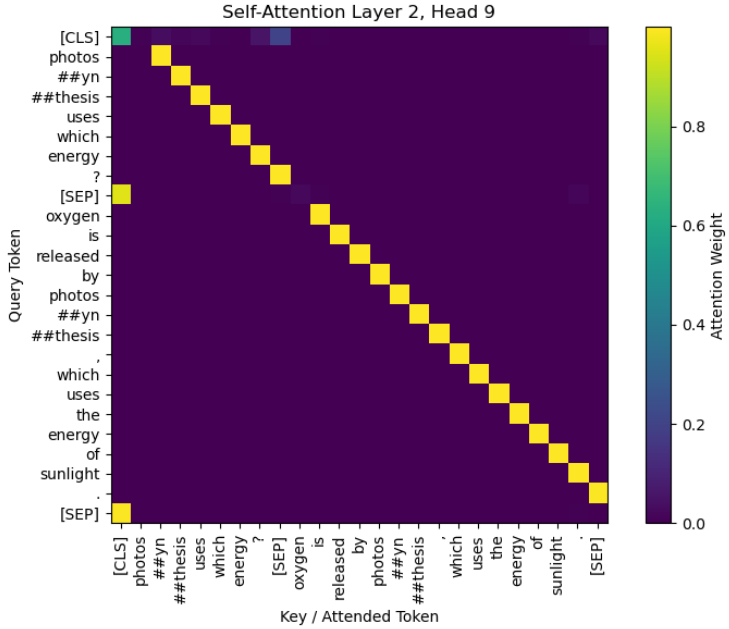

Fig. 9: Layer 2, Head 9

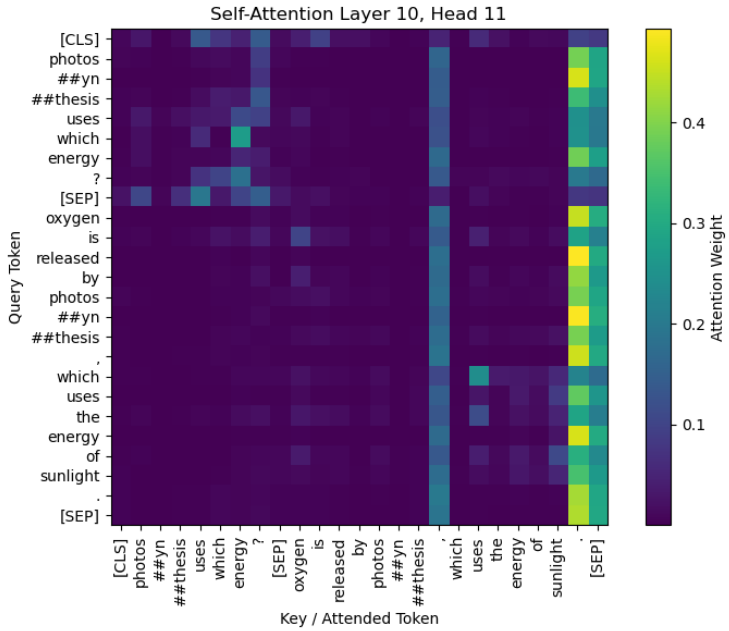

Fig. 10: Layer 10, Head 11