Supplementary Material of Training-free Video Temporal Grounding using Large-scale Pre-trained Models

Minghang Zheng[1], Xinhao Cai[1], Qingchao Chen[2], Yuxin Peng[1], and Yang Liu[1,3⋆]

[1] Wangxuan Institute of Computer Technology, Peking University
[2] National Institute of Health Data Science, Peking University
[3] State Key Laboratory of General Artificial Intelligence, Peking University
{minghang,qingchao.chen,pengyuxin,yangliu}@pku.edu.cn
xinhao.cai@stu.pku.edu.cn

This supplementary material provides additional information and visualization relating to our method. In Section 1, we provide more implementation details about the LLM prompt. In Section 2, we provide more visualization results.

1 Implementation Details of LLM Prompting

We propose the LLM prompting to enable LLMs to analyze query texts, identify multiple potential events therein, and analyze the order and relationship of these events. Specifically, we request the large language model to:

- Reasoning: Analyze the user's query and infer the sub-events it may contain.
- Order of sub-events: Provide each sub-events in chronological order.
- Relationships between sub-events: Consider three types of relationships: Single event, simultaneously (i.e. the sub-events occur simultaneously), and sequentially (i.e. the sub-events occur sequentially).
- Textual descriptions: Generate text descriptions for each sub-event. Due to the uncertainty in LLM output, we request multiple descriptions for each event to enhance the stability.

We require the LLMs to output in the JSON format and provide LLMs an exemplar for in-context learning. The complete prompt is:

LLM prompt

The user will input a text query describing human activities in a video. Your task is to analyze the user's query and break it down into one or more sub-queries, each of which describes only a single simple action. Then, you need to provide multiple textual descriptions for each sub-query as comprehensively as possible. Finally, you need to analyze the temporal relationships between these sub-queries, including whether

⋆ Corresponding author

they occur simultaneously or sequentially, and the order in which they occur.

You should only respond in JSON format as described below:

INSTRUCTIONS OF OUTPUTS:

Your outputs should contain "query_json": "<query_json>". The query json should be structured as follows:
- Translate user descriptions into JSON-compatible format.
- Split the user query into one or more sub-queries. Make sure the sub-queries only describe simple human action. If there is only a simple action descript in the user's query, no need to split multiple sub-queries.
- You need to sort these sub-queries in the order that they are likely to occur in the video.
- Re-write the original query and generate descriptions for each sub-query. You can diversify the sentence structure and word usage, but you should strictly keep the same semantic meaning. Do not add uncertain details that do not associate with the target video. The rewriting should strictly follow the factual information in the original query",
- "sub_query_id" = 0 represents the re-writted original query.
Example:

User Input: "a person is sitting in front of a computer sneezing."

query_json = [
{
"sub_query_id": 0,
"descriptions": ["An individual is seated at a desk, sneezing while using a computer.", "Someone is in front of a computer, sneezing as they sit.", "A person sneezes while sitting in front of their computer."]
},
{
"sub_query_id": 1,
"descriptions": ["A person is sitting in front of a computer.", "Someone is positioned facing a computer.", "Someone is seated before a monitor, interacting with a computer."]
},
{
"sub_query_id": 2,
"descriptions": ["A person is sneezing.", "A sneezing person", "A person starts sneezing."]
}
]

This output must be compatible with Python's json.loads() function.

RESPONSE TEMPLATE:
{
"reasoning": "reasoning",
"sub-queries": "a list of sub-queries for the Temporal Localizer, sorted by the order that they are likely to occur in the video",
"relationship": "The temporal relationships between these sub-queries",
"query_json": "<query_json>"
}

DEFINITION OF RELATIONSHIP BETWEEN SUB-QUERIES:
In your response, the relationship can only be one of the following three choices:
1. single-query: there is only one sub-query.
2. simultaneously: the events corresponding to the sub-queries should occur simultaneously.
3. sequentially: the events corresponding to the sub-queries should occur sequentially.

Example:
{
"reasoning": "The user described a person who is sitting in front of a computer and sneezing. There are two actions in the user's query: sitting and sneezing. To locate the most relevant video segment, I would search for videos containing scenes where these actions occur simultaneously.",
"sub-queries": "1. A person is sitting in front of a computer. 2. A person is sneezing.",
"relationship": "simultaneously",
"query_json" : [
{
"sub_query_id": 0,
"descriptions": ["An individual is seated at a desk, sneezing while using a computer.", "Someone is in front of a computer, sneezing as they sit.", "A person sneezes while sitting in front of their computer."]
},
{
"sub_query_id": 1,
"descriptions": ["A person is sitting in front of a computer.", "Someone is positioned facing a computer.", "Someone is seated before a monitor, interacting with a computer."]
},

{
"sub_query_id": 2,
"descriptions": ["A person is sneezing.", "A sneezing person", "A person starts sneezing."]
}
] }
Again, your response should be in JSON format and can be parsed by Python json.loads(). Please provide the output in JSON format directly, without any additional context. Rewrite the original query when "sub_query_id" = 0.

2 Qualitative Results

As shown in Figures 1 and 2, we visualize the output of the LLM. As can be seen, when the query contains multiple events, the LLM can understand and analyze whether these events occur simultaneously or sequentially, and give the possible order of these events and a text description of each event.

In Figure 3, we show the visualization results of the VLM localizer, including the visual-text similarity, the dynamic and static parts predicted by the model, and the complete prediction. It can be found that if only the proposal with the highest similarity is selected, i.e., the static part, the predicted events are incomplete. On the other hand, our dynamic scoring models the dynamic transition between events, helping the VLM localizer locate the complete events.

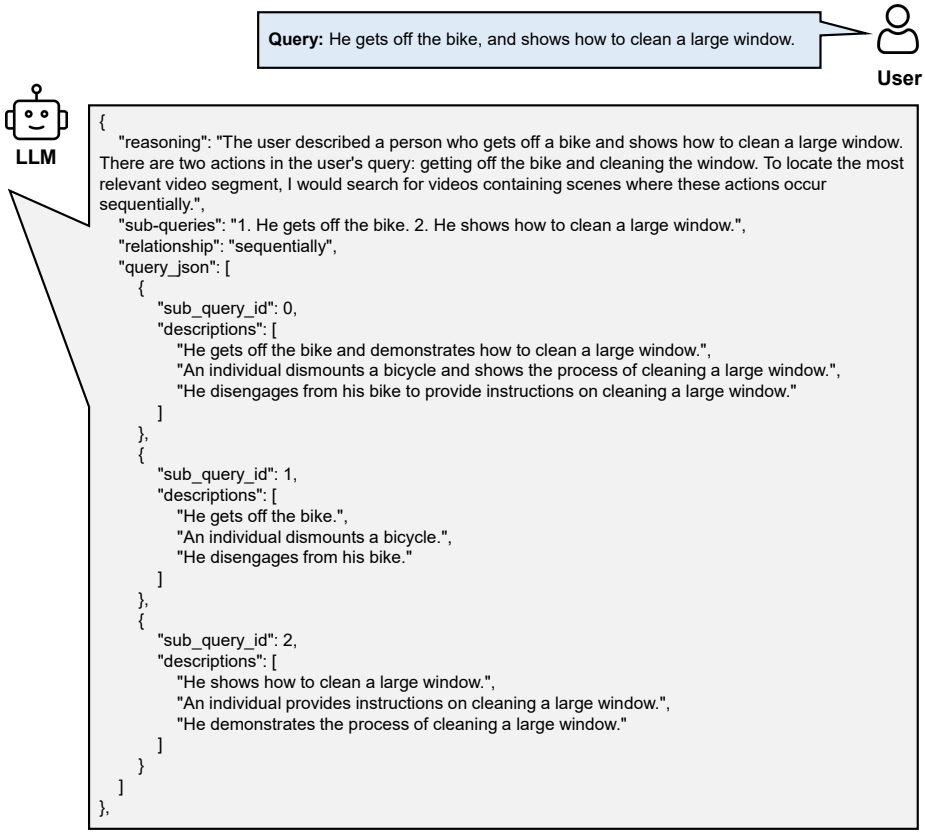

Fig. 1: Visualization of the outputs of LLM on the ActivityNet dataset.

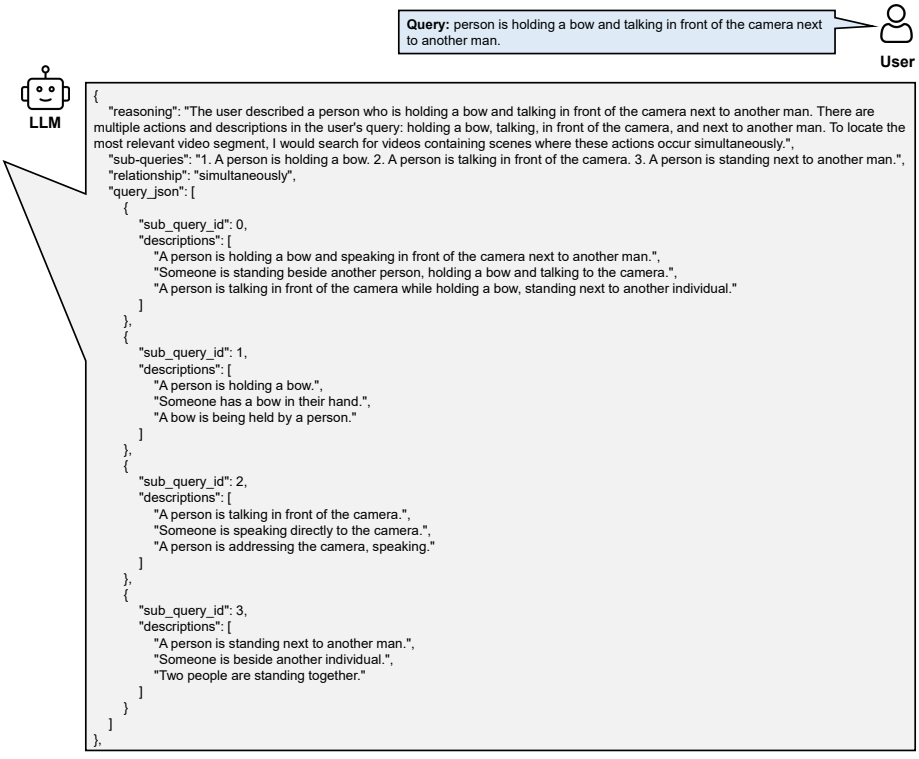

Fig. 2: Visualization of the outputs of LLM on the ActivityNet dataset.

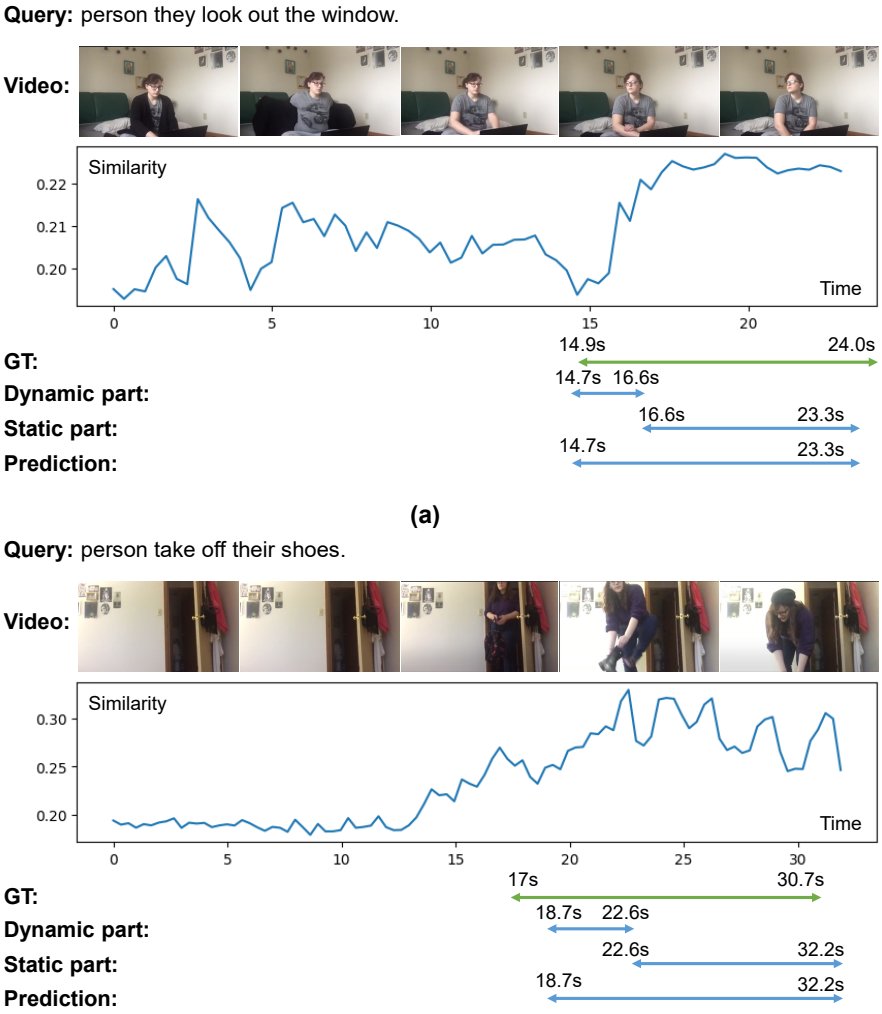

Fig. 3: Visualization of the outputs of VLM localizer on the Charades dataset.