Supplementary Material for "Event Graph Guided Compositional Spatial-Temporal Reasoning for Video Question Answering"

Ziyi Bai, Ruiping Wang, *Senior Member, IEEE*, Difei Gao, *Member, IEEE*, and Xilin Chen, *Fellow, IEEE*

Overview

In this supplementary document, we introduce in detail the graph search algorithm (Section I) and visualize more cases of the constructed event graphs together with the QA results (Section II).

I. Details in Graph Search Algorithm

In our method, we use order embeddings to preserve the hierarchical and temporal order among nodes in the event graph. We propose an improved graph search algorithm to generate such order embeddings. To help understand the details of the algorithm, we visualized the complete graph search process through a simple example and provided algorithm pseudo-code.

A. An Example of Graph Search Algorithm.

As shown in Figure 1, given an event graph containing 28 nodes with their corresponding timestamp t and confidence score c of the video clip on the left, we apply the improved DFS graph search algorithm to number each node. First, a virtual event node R is added at the top of the graph as the start node for graph search. It is connected to all action nodes. Then, the isolated nodes are connected to the graph to guarantee the connectivity of the graph. At last, we apply the improved DFS algorithm to the connected graph. In this way, each node is assigned a unique search order which will be used to generate the order embedding.

B. Pseudo-Code of Graph Search Algorithm.

Algorithm 1 shows the detailed procedure to produce the search order sequence $U = [u_1, u_2, ..., u_{L_V}]$ of an event graph based on the classical Depth-First Search, where L_V is the number of nodes.

Algorithm 1 Graph Search Algorithm

```
Input: Event Graph $\mathcal{G}_{evt} = (\mathcal{V}, \mathcal{E})$; Virtual event node $R$;
Output: Search Order Sequence $U$
 1: Visited-Nodes $\mathcal{S} = [\,]$; Search Order $U = [\,]$
 2: $\mathcal{S}$ = DFS($R$)
 3: for $v$ in $\mathcal{V}$ do
 4:     $U.append(\mathcal{S}.index(v))$    ▷ $index(.)$ can return the index of the node $v$ in the list $\mathcal{S}$
 5: end for
 6: return $U$

 7: function DFS($V$)
 8:     Time List $\mathcal{T} = [\,]$; Score List $\mathcal{C} = [\,]$
 9:     for $v$ in $V$.child do
10:         $\mathcal{T}.append(v.\text{time})$
11:         $\mathcal{C}.append(v.\text{score})$
12:     end for
13:     $V'$ = SORTED($V.child, \mathcal{T}, \mathcal{C}$)
14:     for $v$ in $V'$ do
15:         DFS($v$)
16:     end for
17:     if $V$ not in $\mathcal{S}$ then
18:         $\mathcal{S}.append(V)$
19:     end if
20:     return $\mathcal{S}$
21: end function

22: function SORTED($V, \mathcal{T}, \mathcal{C}$)
23:     if $\mathcal{T}$.min == $\mathcal{T}$.max then
24:         return $V.sortedBy(\mathcal{C})$    ▷ Sort in descending order by score
25:     else
26:         return $V.sortedBy(\mathcal{T})$    ▷ Sort in ascending order by time
27:     end if
28: end function
```

II. Extended Quantitative Experiments

We show more visualization of the generated event graphs together with the QA results predicted by our method in Figure 2 and 3. All the videos are from the AGQA dataset [1]. For each video example, we visualize the top-3 action proposals with their timestamps shown as a green bar. The key frames are arranged in time order as follows. Then two frames that are related to the questions are chosen to visualize the object detection and scene graph generation results. We show the top-5 objects and relationships of each scene separately. At last, for each video, we select some representative QA pairs with predicted answers from our methods.

As shown in Figure 2, our model answered all 6 questions correctly which are of different types from querying the local object to global action, from binary answer type to open answer type, from step 2 to step 7, *etc*. Due to our method can represent visual concepts of different levels in the video,

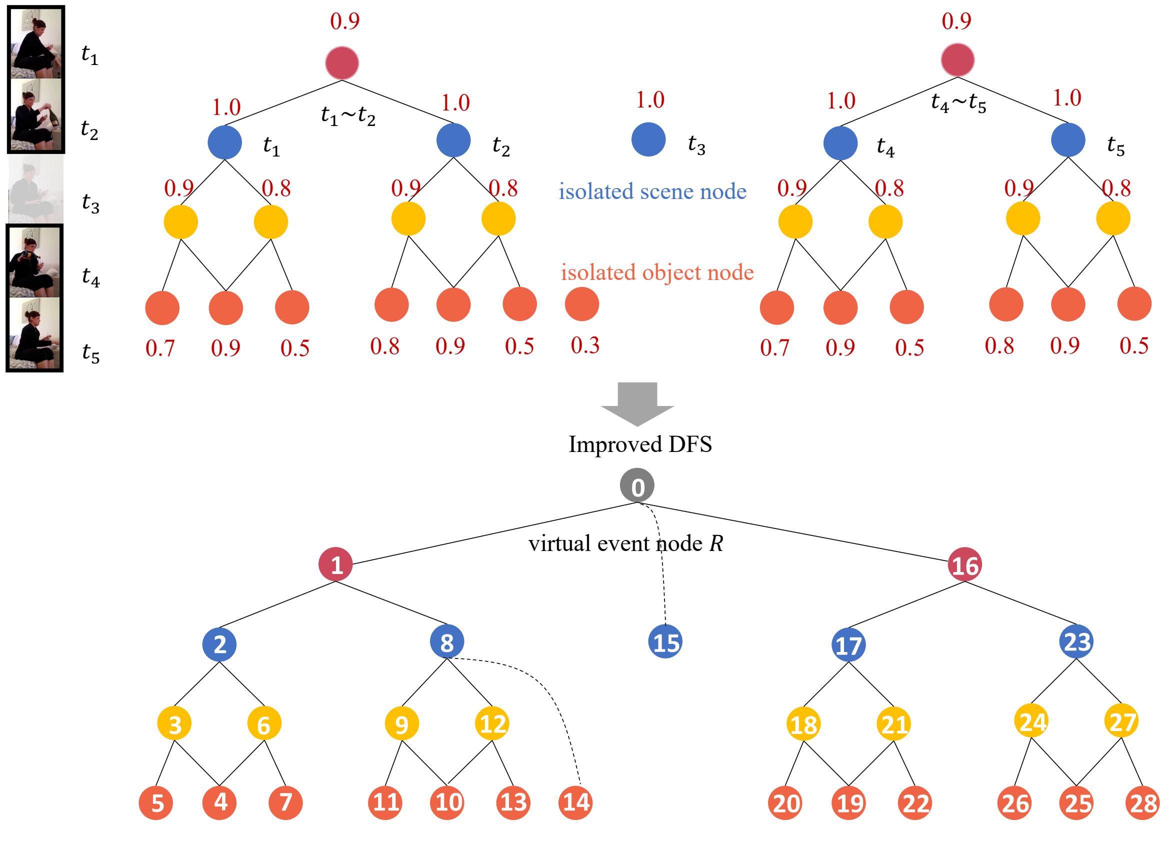
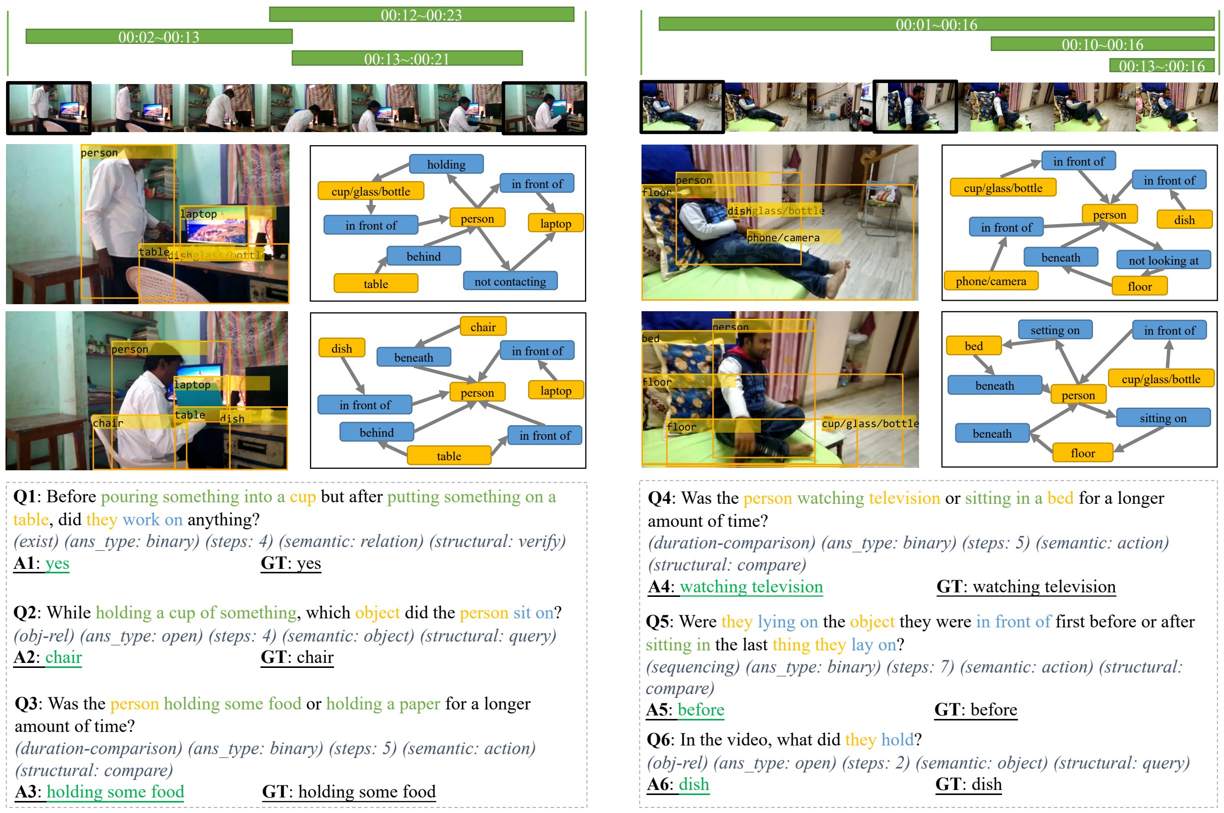


Fig. 1: An example of graph serialization process using extended DFS on an event graph with 28 nodes.



Q1: Before pouring something into a cup but after putting something on a table, did they work on anything?
(exist) (ans_type: binary) (steps: 4) (semantic: relation) (structural: verify)
A1: yes **GT**: yes

Q2: While holding a cup of something, which object did the person sit on?
(obj-rel) (ans_type: open) (steps: 4) (semantic: object) (structural: query)
A2: chair **GT**: chair

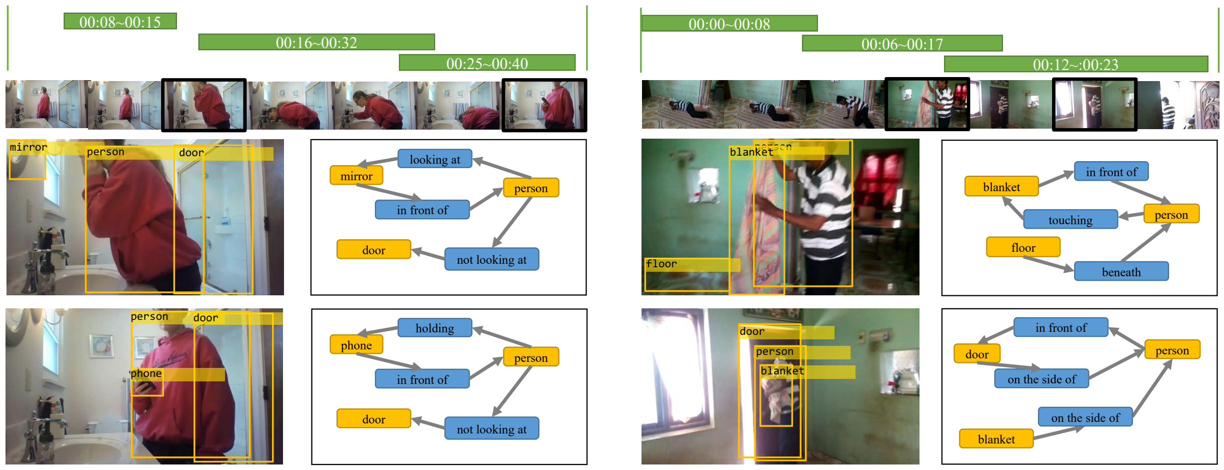
Q3: Was the person holding some food or holding a paper for a longer amount of time?
(duration-comparison) (ans_type: binary) (steps: 5) (semantic: action) (structural: compare)
A3: holding some food **GT**: holding some food

Q4: Was the person watching television or sitting in a bed for a longer amount of time?
(duration-comparison) (ans_type: binary) (steps: 5) (semantic: action) (structural: compare)
A4: watching television **GT**: watching television

Q5: Were they lying on the object they were in front of first before or after sitting in the last thing they lay on?
(sequencing) (ans_type: binary) (steps: 7) (semantic: action) (structural: compare)
A5: before **GT**: before

Q6: In the video, what did they hold?
(obj-rel) (ans_type: open) (steps: 2) (semantic: object) (structural: query)
A6: dish **GT**: dish

Fig. 2: Success QA cases of different question types with various reasoning steps.



Q7: What did they touch while holding the object they were taking first?
(obj-rel) (ans_type: open) (steps: 5) (semantic: object) (structural: query)
A7: phone **GT**: dish

Q8: What was the person interacting with before holding the object they were taking first but after watching outside of the window?
(rel-act) (ans_type: open) (steps: 5) (semantic: object) (structural: query)
A8: phone **GT**: mirror

Q9: Was the person washing their hands before or after holding the last thing they touched?
(sequencing) (ans_type: binary) (steps: 6) (semantic: action) (structural: compare)
A9: after **GT**: before

Q10: Before opening a door but after lying on the floor, was a blanket the first thing they stood on?
(superlative/obj-rel) (ans_type: binary) (steps: 4) (semantic: object) (structural: verify)
A10: no **GT**: yes

Q11: Which did they go in front of after awakening in bed, a blanket or the object they were in last?
(obj-rel) (ans_type: binary) (steps: 3) (semantic: object) (structural: choose)
A11: door **GT**: doorway

Q12: While standing up, of everything the person sat on, was the first the object they were behind last?
(superlative/obj-rel) (ans_type: binary) (steps: 7) (semantic: object) (structural: query)
A12: no **GT**: yes

Fig. 3: Different types of failure QA cases.

and organize them with the structure of the event graph to preserve their original spatial-temporal relationship, the model can effectively establish the correspondence between visual concepts in the event graph and semantic words in the questions, which facilitate the VideoQA tasks.

On the contrary, the model fails in some cases, which are shown in Figure 3. These failure cases fall into two categories. 1) The construction of the event graph is accurate, but the model inference is incorrect (the cases on the left). 2) The construction of the event graph is incorrect or incomplete, and the model inference is misled (the case on the right).

Specifically, in Q7 to Q9, the model gave wrong answers due to the complex indirect reference, "the object..." and "the last thing...". As for the right side cases, the model cannot capture the key relation `stand on` in Q10, and cannot distinguish between `door` and the `doorway` in Q11. In Q12, the incomplete event graph and indirect reference can both lead to failure.

References

[1] M. Grunde-McLaughlin, R. Krishna, and M. Agrawala, "Agqa: A benchmark for compositional spatio-temporal reasoning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 11 287–11 297.