Unsupervised Episode Detection for Scene Graph Understanding

Problem: Flat Scene Graphs Lack Structure

Scene graphs represent video actions as sequences of triplets:

`[person, verb, pick-up], [pick-up, dobj, mop], [mop, from, floor], ...`

Challenge: Temporal reasoning requires understanding **episode boundaries**

Without hierarchy:

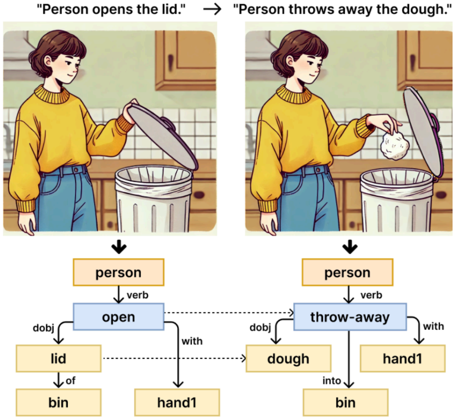
- Flat sequence of 11 actions
- No phase structure
- Model must infer relationships

With hierarchy:

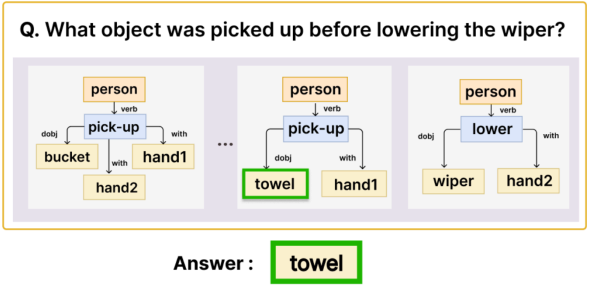
- Episode 1: Prepare & sweep
- Episode 2: Store supplies
- Episode 3: Retrieve from cabinet

Key Question: How to detect meaningful phases *without supervision*?

Video Scene Graph



SGQA Task

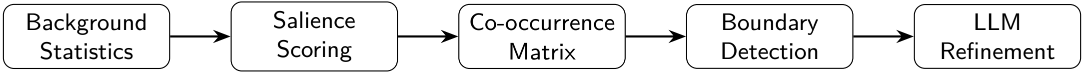


Answer temporal reasoning questions:
“What was done *before* X?”, “After completing A...”

Structure: **Agent** $\xrightarrow{\text{verb}}$ **Action** $\xrightarrow{\text{dobj}}$ **Object**
with `with`, `of`, `into` relations

Our Approach: EpiMine-inspired Episode Detection for Scene Graphs

Key Insight: Episode boundaries occur where **co-occurrence patterns shift**



Hybrid approach:

- **Statistical**: Unsupervised boundary detection
- **Semantic**: LLM generates episode names/descriptions

Step 1: Salience – Identifying Discriminative Terms

Goal: Find terms that characterize the current activity

Formula:

$$\text{salience}(t) = \underbrace{(1 + \log(f_{\text{fg}})^2)}_{\text{foreground boost}} \times \underbrace{\log\left(\frac{N_{\text{bg}}}{f_{\text{bg}}}\right)}_{\text{IDF component}}$$

- f_{fg}: term frequency in current sample (foreground)
- f_{bg}: term frequency across all samples (background)

Example (Car Cleaning):

Term	Salience	Interpretation
mop-stick	18.86	Highly discriminative
wall	9.98	Discriminative
person	≈ 0	Common (filtered)

Tunable: `min_freq`

Filter terms with $f_{\text{fg}} <$ `min_freq`

Higher $\rightarrow$ fewer episodes
Optimal: `min_freq=2`

Step 2: Co-occurrence Matrix

Goal: Measure similarity between consecutive actions using top-k salient terms

Jaccard Similarity (using only top-k key terms):

$$\text{cooccur}(A_i, A_j) = \frac{|\text{terms}_k(A_i) \cap \text{terms}_k(A_j)|}{|\text{terms}_k(A_i) \cup \text{terms}_k(A_j)|}$$

Intuition:

- **High co-occurrence** $\rightarrow$ same episode
- **Low co-occurrence** $\rightarrow$ boundary

$$A_0 \xrightarrow{0.67} A_1 \xrightarrow{0.46} \cdots \underset{\text{boundary!}}{\xrightarrow{0.20}} A_6$$

Tunable: `top_k`

Limit to top-k salient terms

`topk=all`: noisy (18.3% avg)
`topk=5`: too sparse (20.8%)
Optimal: `topk=10` (24.2%)

Step 3: Boundary Detection

Algorithm:

1. Compute consecutive scores: $s_i = \text{cooccur}(A_i, A_{i+1})$
2. Calculate threshold: $\tau = \mu - t \cdot \sigma$
3. If $s_i < \tau$: start new episode

Tunable: `threshold_std`

Formula: $\tau = \mu - t \cdot \sigma$

Lower $t \to$ more boundaries
Higher $t \to$ fewer episodes

t	Avg	Max
1.0	15.8%	30%
1.5	**24.2%**	**50%**
2.0	23.3%	30%

Example (11 actions):

$A_0 \to A_1$	...	$A_5 \to A_6$	...	$A_9 \to A_{10}$
0.67	...	**0.20**	...	**0.31**

$\tau = 0.437 - 1.0 \times 0.129 = 0.308$

Detected Episodes:

- Episode 0: Actions [0-5] – pick-up, sweep, close, place, open, put
- Episode 1: Actions [6-9] – move, open, pick-up, move
- Episode 2: Action [10] – pick

Step 4: LLM Refinement

LLM refines boundaries into structured episodes:


```
{
  "episode_id": 0,
  "name": "Prepare and Sweep",
  "description": "Pick up mop,
    sweep floor, manage door",
  "core_structure": {
    "agent": "person",
    "actions": ["pick-up","sweep"],
    "objects": ["mop-stick","floor"]
  },
  "time": {
    "action_indices": [0,1,2,3,4,5],
    "duration": 6
  }
}
```


Tunable: `use_llm`, `gen_model`

`use_llm=0`: structural only
`use_llm=1`: semantic naming

`gen_model`: gpt5-mini vs gpt5

Setting	Avg	Max
noLLM	18.9%	30%
LLM	**23.3%**	**50%**

+4.4% avg improvement

Output: Episode name, description, core structure, temporal info, discriminative terms

Experiments: Hard-10 Benchmark

Motivation: General SGQA accuracy (88.6%) may hide weaknesses on hard cases

Hard-10 Construction:

1. Run GPT-5 on full SGQA (500 questions)
2. Identify **88 questions** where GPT-5 failed
3. Select **first 10** for hyperparameter tuning

Hard Question Categories:

Category	Count
temporal_ordering	57
multi_step	17
both_hands	9

Grid Search

$3 \times 2 \times 3 \times 2 = 36$ configs

threshold_std	1.0, 1.5, 2.0
min_freq	1, 2
top_k	5, 10, all
use_llm	0, 1

Baseline: 20% (2/10)

Results: Grid Search on Hard-10

Best Configurations:

Acc	t	**mf**	**topk**	**LLM**
50%	1.5	2	10	Yes
40%	1.5	2	5	Yes
30%	various	various	5,10	mixed
20%	Baseline (no episodes)			

Key findings:

- **+150% relative improvement** (20% → 50%)
- Top configs share: t=1.5, mf=2, LLM=yes
- Parameter interactions matter

Parameter Impact:

Parameter	**Best**	**Why**
t	1.5	Moderate sensitivity
mf	2	Filter noise
topk	10	Focus key terms
LLM	yes	+4.4% avg

topk ablation

all	18.3%	noisy
5	20.8%	sparse
10	**24.2%**	balanced

Results: Full SGQA & Analysis

Full SGQA (500 questions)

Method	EM	Δ
Baseline	88.6%	–
EpiMine (gpt5-mini)	89.6%	+1.0%
EpiMine (gpt5)	**90.6%**	+2.0%

Hard-10 (10 questions)

Method	EM	Δ
Baseline	20%	–
EpiMine (gpt5-mini)	**50%**	+30%
EpiMine (gpt5)	40%	+20%

LLM Impact Analysis:

Config	noLLM	+LLM	Δ
t=1.5, topk=10	20%	**50%**	+30%
t=1.5, topk=5	10%	**40%**	+30%
t=1.0, topk=10	30%	10%	-20%

Insight: LLM amplifies good boundaries, but can hurt poor segmentation

Takeaway: Larger gains on harder questions; model choice is task-dependent