Hierarchical Semantic Segmentation with Autoregressive Language Modeling

Supplementary Material

This document supplements the main paper with the following content:

1. Proof showing our proposed captions belong to a context-free language. (supplements **Section 3**)
2. Issues with LLM-as-a-Judge metric for our task. (supplements **Section 4.1**)
3. Breakdown of category-level performance. (supplements **Section 4.2**)
4. Additional qualitative results. (supplements **Section 4.2**)

1. Captions as Context-Free Language

We show that the language, L, for our proposed hierarchical captions is context-free and thus machine-comprehensible [3]. To do this, we construct a context-free grammar (CFG) consisting of terminal symbols Σ, non-terminal symbols V, productions rules R, and a start symbol S, that generates all valid strings in L.

Language Definition. The language L consists of strings that describe the decomposition of an object into its components using special tags with indices representing the positions of entities, proceeding breadth-first to the subpart level. For readability, we omit standard mask-as-embedding tags: <SEG>, , and .

Each string in L follows the general format: [OBJECT 1] <D$_1$> [OBJECT 2] <D$_2$> ... [OBJECT N] <D$_N$> . <C$_1$> [OBJECT 1 PART 1] ... <C$_N$> [OBJECT N PART M SUBPART K] ... </D$_{N+M}$> ... </D$_1$>, where N is the number of objects, M is the number of parts, and K is the number of subparts. The tags <D$_i$> and </D$_i$> denote the start and end of the decomposition of the i-th entity and <C$_i$> denote the content (children) of the i-th entity.

Assumptions. We assume that indices for our special tokens are bounded by the maximum number of nodes present in a given training dataset. This assumption aligns with practical implementations where the tokenizer for a language model must be aware of all special tokens a priori to correctly parse them. Without this boundedness, the language would become context-sensitive due to the possibility of an unbounded number of entities. Specifically, production rules could allow a single non-terminal symbol (the left-hand side) to be replaced by an arbitrarily long string of symbols (the right-hand side). This unrestricted growth would require the grammar to maintain an infinite number of states or counts, which exceeds the capabilities of a context-free grammar.

Context-Free Grammar Construction. We now construct the context-free grammar $G = (V, \Sigma, R, S)$ by defining our symbols and production rules.

Terminal Symbols (Σ). The set of terminal symbols includes: Σ = {[OBJECT 1], [OBJECT i], [OBJECT i PART j], [OBJECT i PART j SUBPART k], <D$_l$>, </D$_l$>, <C$_m$>, .}, where

- $i \in \{1 \dots, N\}$ (object indices)
- $j \in \{1, \dots, M\}$ (part indices)
- $k \in \{1, \dots, K\}$ (subpart indices)
- $l, m \in \{1, \dots, N+M\}$ (decomposition indices/children)

Each tag, along with its specific index, is treated as a unique terminal symbol. This approach ensures that opening and closing tags (e.g., <D$_i$> and </D$_i$>) are correctly matched without requiring cross-referencing mechanisms, which are beyond the capabilities of a context-free grammar (CFG). We define our rules such that children are always derived in the context of their parents, implicitly maintaining the associating between indices.

Non-terminal Symbols (V). The set of non-terminal symbols includes: $V = \{S,\ D_i,\ C_i,\ E_i \mid i = 1, 2, \dots, N + M\}$, where S is the start symbol, D_i represents the decomposition of the i-th entity, C_i represents the content (child) of the i-th entity, and E_i represents the entity content (e.g., children) of the i-th entity.

Production Rules (R). We define the production rules, for each entity index $i \in \{1, \dots, M + N\}$, as:

1. **Start Rule:**

$$S \rightarrow \texttt{[OBJECT 1]}\ D_1$$

2. **Decomposition Rule:**

$$D_i \rightarrow \texttt{<D}_i\texttt{>}\ C_i D_{\text{next}(i)}\ \texttt{</D}_i\texttt{>}$$

where $D_{\text{next}(i)}$ handles the decomposition of the next entity. For breadth-first traversal, we define:

$$D_{\text{next}(i)} = \begin{cases} D_{i+1}\texttt{<D}_i\texttt{>}, & \text{if } i < N + M \\ \epsilon, & \text{if } i \geq N + M \end{cases}$$

where ϵ is the empty string. This creates a sequence where each entity at the current level is expanded before moving on to entities at a deeper level. The recursive nature of $D_{\text{next}(i)}$ allows the CFG to handle all entities at a given breadth level before descending to the parts or subparts within those entities. This rule ensures that

each layer of the hierarchy is processed in sequence, enforcing the breadth-first property.

3. **Content Rule:** For each $i \in \{1, 2, \ldots, N+M\}$:

$$C_i \rightarrow \texttt{<C}_i\texttt{>}(E_i)^* \mid \epsilon$$

Each $\texttt{<C}_i\texttt{>}$ acts as an independent symbol that can recursively expand into its corresponding child entities. The CFG implicitly manages these expansions without needing cross-referencing, as the recursive nature of the production rules ensures that all entities sharing the same <C> tag are parsed in sequence.

4. **Entity Content Rules:**
For object entities, we define:

$$E_i \rightarrow \texttt{[OBJECT } i \texttt{ PART } j\texttt{]} \mid \texttt{[OBJECT } i \texttt{ PART } j\texttt{]}\, E_i$$

and for (sub)part entities:

$$E_i \rightarrow \texttt{[OBJECT } i \texttt{ PART } j \texttt{ SUBPART } k\texttt{]} \mid \texttt{[OBJECT } i \texttt{ PART } j \texttt{ SUBPART } k\texttt{]}\, E_i$$

This rule allows for the recursive decomposition of entities (e.g., parts into subparts).

5. **Terminal Entities:** For entities with no further decomposition (e.g., subparts):

$$E_{N+M+k} \rightarrow \epsilon, \quad D_{N+M+k} \rightarrow \epsilon$$

for $k \in \{1, 2, \ldots, K\}$, where ϵ represent the empty string. These rules represent the leaf nodes of the hierarchy, ending the decomposition.

Example. We now show a simple example of our grammar and rules using a hierarchy with 1 object, 2 parts, and a single subpart. The target string, L, for this sequence is "[OBJECT 1] <D_1> <C_1> [OBJECT 1 PART 1] [OBJECT 1 PART 2]<D_2> <C_2> [OBJECT 1 PART 1 SUBPART 1] </D_2> </D_1>".

To determine if L can be constructed by our CFG, we first start with our start string, [OBJECT 1] and expand based on our production rules:

1. [OBJECT 1] D_1
2. [OBJECT 1] <D_1> C_1 </D_1> D_2
3. [OBJECT 1] <D_1> <C_1> E_1* <C_2> D_2 </D_1>
4. [OBJECT 1] <D_1> <C_1> [OBJECT 1 PART 1] [OBJECT 1 PART 2] D_2 </D_1>
5. [OBJECT 1] <D_1> <C_1> [OBJECT 1 PART 1] [OBJECT 1 PART 2] <D_2> C_2 </D_2> ϵ </D_1>
6. [OBJECT 1] <D_1> <C_1> [OBJECT 1 PART 1] [OBJECT 1 PART 2] <D_2> <C_2> E_2* </C_2> </D_2> ϵ </D_1>
7. [OBJECT 1] <D_1> <C_1> [OBJECT 1 PART 1] [OBJECT 1 PART 2] <D_2> <C_2> [OBJECT 1 PART 1 SUBPART 1] </C_2> </D_2> ϵ </D_1>
8. [OBJECT 1] <D_1> <C_1> [OBJECT 1 PART 1] [OBJECT 1 PART 2] <D_2> <C_2> [OBJECT 1 PART 1 SUBPART 1] </D_2> </D_1>

which matches our target string.

Conclusion. By constructing the context-free grammar $G = (V, \Sigma, R, S)$ that generates the language L, we have shown that L is context-free and capable of producing hierarchical structures with nested tags.

2. LLM-as-a-Judge Metric

We initially attempted to use the LLM-as-a-judge [5] evaluation metric to compare predicted and ground truth captions, as it has been shown to align better with human judgments than traditional metrics like METEOR [1]. However, we found it to be unsuitable for our novel task. Specifically, using the proposed open-source model Llama [4], we found it could not classify outputs as partially correct. Rather, it tended to classify predictions as either fully correct—even if entities were missing—or fully incorrect, even when only one entity was missing. A potential reason for this is the lack of hierarchical relationships present in the training data of these LLMs [2]. This lack of understanding caused the LLM to be unsuitable for evaluating captions in our task.

3. Qualitative Results

Additional quantitative results for HALLUMI are presented in **Figure** 1. We observe that segmentation quality remains relatively stable as the number of entities within a hierarchy increases. This reflect findings from the main paper that caption length does not have a significant impact on segmentation performance. However, the rightmost hierarchy illustrates a failure case where HALLUMI misclassifies the windshield as the window, a closely related entity. Future work could address this limitation by ensuring that segmentations at a given hierarchy level do not share pixels. This could potentially be achieved through a loss function that penalizes overlap between entities within the same level.

4. Fine-grained Category-level Performance

A breakdown of the category-level performance results from **Section 4.2** is provided in **Table** 1. For this analysis, we group common subparts and use a Kruskal-Wallis test to assess differences in the distribution of IoU values. For most groupings, we find no significant difference in intra-group IoU distributions. However, in 14 out of 38 groupings, there is significant evidence that at least one intra-group IoU distribution differs from the rest. This supports our conclusion that shared subpart groupings are represented with varying degrees of performance.

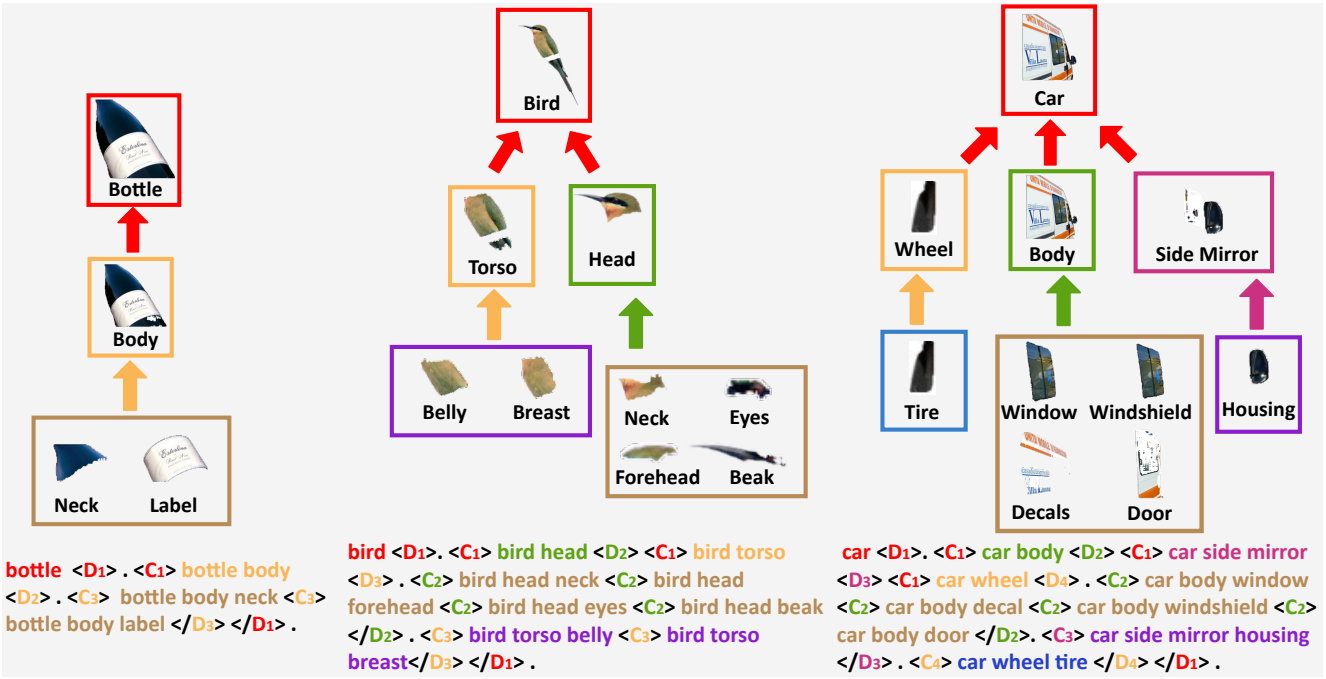

Figure 1. Results from HALLUMI on hierarchies with few entities (left), a moderate number of entities (center), and a large number of entities (right). Segmentation quality remains consistent as the number of entities increases. The right figure illustrates a failure case where HALLUMI misclassifies the windshield as the closely related window.

Metric	Cheek	Forehead	Eyes	Nostrils	Back	Lobe	Fins	Mouth	Neck	Side	Surface	Tire	Fender	Rim	Cap	Snout	Belly	Beam	Nose	Ears
# Groups	3	5	6	3	5	2	3	5	8	3	2	2	2	3	3	2	3	2	2	3
$p < \alpha$	✗	✓	✓	✓	✓	✗	✗	✓	✓	✗	✗	✓	✗	✓	✓	✗	✓	✓	✗	✗
Metric	Area	Toes	Wrist/Ankle	Shank/Forearm	Fork	Tube	Arm	Shoulders	Chest	Heel	Window	Hood	Windshield	Light	Knee	Decals	Stabilizer	Claws		
# Groups	2	3	2	3	2	4	3	3	2	2	2	2	3	2	2	3	2	2		
$p < \alpha$	✗	✗	✓	✓	✗	✗	✗	✓	✓	✗	✓	✗	✗	✗	✓	✗	✓	✓		

Table 1. Category-level performance results with statistical significance from a Kruskal-Wallis test. We observe that 14 out of 38 subpart groupings have at least one significantly different IoU distribution. We use $\alpha = 0.05$ for all tests, with a Bonferonni correction that adjusted α for the number of pairwise comparisons.

References

[1] Chongyan Chen, Mengchen Liu, Noel Codella, Yunsheng Li, Lu Yuan, and Danna Gurari. Fully authentic visual question answering dataset from online communities. *arXiv preprint arXiv:2311.15562*, 2023. 2

[2] Josh Myers-Dean, Jarek Reynolds, Brian Price, Yifei Fan, and Danna Gurari. Spin: Hierarchical segmentation withsubpart granularity innatural images. In *Computer Vision – ECCV 2024*, pages 275–292, Cham, 2025. Springer Nature Switzerland. 2

[3] Lena Strobl, William Merrill, Gail Weiss, David Chiang, and Dana Angluin. What formal languages can transformers express? a survey. *Transactions of the Association for Computational Linguistics*, 12:543–561, 2024. 1

[4] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023. 2

[5] Lianghui Zhu, Xinggang Wang, and Xinlong Wang. Judgelm: Fine-tuned large language models are scalable judges. *arXiv preprint arXiv:2310.17631*, 2023. 2