LLM4SGG: Large Language Models for Weakly Supervised Scene Graph Generation

Supplementary Material

A. Details of Method

A.1. Details of Prompt

We provide the complete prompts for extracting triplets from paraphrased captions (Table 5) and from original captions (Table 6). Moreover, we provide those for aligning entity classes (Table 7), and predicate classes (Table 8) with entity and predicate classes in the target data, respectively.

Task Description

From the given sentence, the task is to extract meaningful triplets formed as <subject, predicate, object>.
To extract meaningful triplets from the sentence, please follow the following two steps.
Step 1: Paraphrase the sentence.
Step 2: From the paraphrased sentence obtained in the Step 1, extract meaningful triplets formed as <subject, predicate, object>.
Note that the subject is the entity or noun that performs the action or is being described, and the object is the entity
or noun that is affected by the action or is receiving the action. The predicate is a verb or adjective without auxiliary verb.

In-context Examples

Let's take a few examples to understand how to extract meaningful triplets.
Question: Given the sentence "a slice of bread is covered with a sour cream and guacamole," extract meaningful triplets.
Answer: Step 1: The sentence can be paraphrased as:
A piece of bread is topped with both sour cream and guacamole.
Step 2: Meaningful triplets, where the subject and object are the simple noun, extracted from the paraphrased sentence are:
<bread, topped with, sour cream>, <bread, topped with, guacamole>.
The meaningful triplets are *<bread, topped with, sour cream>, and <bread, topped with, guacamole>*.
Question: Given the sentence "A beautiful woman walking a dog on top of a beach," extract meaningful triplets.
Answer: Step 1: The sentence can be paraphrased as:
A lovely woman strolling with a dog on the beach.
Step 2: Meaningful triplets, where the subject and object are the simple noun, extracted from the paraphrased sentence are:
<woman, strolling with, dog>, <woman, on, beach>, <dog, on, beach>.
The meaningful triplets are *<woman, strolling with, dog>, <woman, on, beach>, and <dog, on, beach>*.
Question: Given the sentence "Four clock sitting on a floor next to a woman's feet," extract meaningful triplets.
Answer: Step 1: The sentence can be paraphrased as:
Four clocks are placed on the floor beside a woman's feet.
Step 2: Meaningful triplets, where the subject and object are the simple noun, extracted from the paraphrased sentence are:
<clocks, placed on, floor>, <clocks, beside, feet>.
The meaningful triplets are *<clocks, placed on, floor> and <clocks, beside, feet>*.
Question: Given the sentence "One person sits in a chair looking at her phone while another rests on the couch,"
extract meaningful triplets.
Answer: Step 1: The sentence can be paraphrased as:
A person is seated in a chair, using their phone, while someone else is relaxing on the couch.
Step 2: Meaningful triplets, where the subject and object are the simple noun, extracted from the paraphrased sentence are:
<person, seated in, chair>, <person, using, phone>, <person, relaxing on, couch>.
The meaningful triplets are *<person, seated in, chair>, <person, using, phone>, and <person, relaxing on, couch>*.
Question: Given the sentence "A lady and a child near a park bench with kites and ducks flying in the sky and on the ground,"
extract meaningful triplets.
Answer: Step 1: The sentence can be paraphrased as:
A woman and a child are close to a park bench, while kites soar through the sky and ducks move around on the ground.
Step 2: Meaningful triplets, where the subject and object are the simple noun, extracted from the paraphrased sentence are:
<woman, close to, park bench>, <child, close to, park bench>, <kites, soar through, sky>, <ducks, move around, ground>.
The meaningful triplets are *<woman, close to, park bench>, <child, close to, park bench>, <kites, soar through, sky>*,
and *<ducks, move around, ground>*.
Question: Given the sentence "Two men sit on a bench near the sidewalk and one of them talks on a cell phone,"
extract meaningful triplets.
Answer: Step 1: The sentence can be paraphrased as:
Two guys are seated on a bench near the road, and one of them talks on a mobile phone.
Step 2: Meaningful triplets, where the subject and object are the simple noun, extracted from the paraphrased sentence are:
<guys, seated on, bench>, <bench, near, road>, <guy, talks on, phone>.
The meaningful triplets are *<guys, seated on, bench>, <bench, near, road>, and <guy, talks on, phone>*.

Actual Question

Question: Given the sentence Input, extract meaningful triplets. Answer:

Table 5. Prompt for triplet extraction from paraphrased caption.

A.2. Aligning entity classes within a larger predefined lexicons

When aligning a query entity with entity classes in the Visual Genome and GQA datasets via an LLM, we list 150 and 200 entity classes in the prompt, respectively. However, as the number of of entity classes increases, we would need to add the entity classes in the prompt, which would eventually reach the maximum prompt length (e.g., 4096 tokens in ChatGPT). This makes it impossible to align a query entity with entity classes in a large predefined lexicon. To address it, as shown in Figure 5, we can construct several prompts in a hierarchical way. More precisely, for 1,594 entity classes in VinVL [52] as an example of a large predefined lexicon,

Task Description

From the given sentence, the task is to extract meaningful triplets formed as <subject, predicate, object>.
Note that the subject is the entity or noun that performs the action or is being described, and the object
is the entity or noun that is affected by the action or is receiving the action. The predicate is a verb or
adjective without auxiliary verb, and is represented without the tense.

In-context Examples

Let's take a few examples to understand how to extract meaningful triplets.
Question: Given the sentence "a slice of bread is covered with a sour cream and guacamole,"
extract the meaningful triplets.
Answer: Meaningful triplets are *<bread, covered with, sour cream>, and <bread, covered with, guacamole>*.
Question: Given the sentence "A beautiful woman walking a dog on top of a beach,"
extract meaningful triplets.
Answer: Meaningful triplets are *<woman, walking with, dog>, <woman, on, beach>, and <dog, on, beach>*.
Question: Given the sentence "Four clock sitting on a floor next to a woman's feet,"
extract the meaningful triplets.
Answer: Meaningful triplets are *<clock, sitting on, floor> and <clock, next to, feet>*.
Question: Given the sentence "One person sits in a chair looking at her phone while another rests on the couch,"
extract meaningful triplets.
Answer: Meaningful triplets are *<person, sits in, chair>, <person, looking at, phone>*,
and *<person, rests on, couch>*.
Question: Given the sentence "A lady and a child near a park bench with kites and ducks flying in the sky and
on the ground," extract meaningful triplets.
Answer: Meaningful triplets are *<lady, near, park bench>, <child, near, park bench>, <kites, flying in sky>*,
and *<ducks, on, ground>*.
Question: Given the sentence "Two men sit on a bench near the sidewalk and one of them talks on a cell phone,"
extract meaningful triplets.
Answer: Meaningful triplets are *<men, sit on, bench>, <bench, near, sidewalk>, and <man, talks on, phone>*.

Actual Question

Question: Given the sentence Input, extract meaningful triplets. Answer:

Table 6. Prompt for triplet extraction from original caption.

Task Description

The predefined entity lexicon containing 150 lexemes is numbered as follows: 1.airplane 2.animal 3.arm 4.bag 5.banana
6.basket 7.beach 8.bear 9.bed 10.bench 11.bike 12.bird 13.board 14.boat 15.book 16.boot 17.bottle 18.bowl 19.box 20.boy
21.branch 22.building 23.bus 24.cabinet 25.cap 26.car 27.cat 28.chair 29.child 30.clock 31.coat 32.counter 33.cow 34.cup
35.curtain 36.desk 37.dog 38.door 39.drawer 40.ear 41.elephant 42.engine 43.eye 44.face 45.fence 46.finger 47.flag 48.flower
49.food 50.fork 51.fruit 52.giraffe 53.girl 54.glass 55.glove 56.guy 57.hair 58.hand 59.handle 60.hat 61.head 62.helmet
63.hill 64.horse 65.house 66.jacket 67.jean 68.kid 69.kite 70.lady 71.lamp 72.laptop 73.leaf 74.leg 75.letter 76.light 77.logo
78.man 79.men 80.motorcycle 81.mountain 82.mouth 83.neck 84.nose 85.number 86.orange 87.pant 88.paper 89.paw 90.people
91.person 92.phone 93.pillow 94.pizza 95.plane 96.plant 97.plate 98.player 99.pole 100.post 101.pot 102.racket 103.railing
104.rock 105.roof 106.room 107.screen 108.seat 109.sheep 110.shelf 111.shirt 112.shoe 113.short 114.sidewalk 115.sign
116.sink 117.skateboard 118.ski 119.skier 120.sneaker 121.snow 122.sock 123.stand 124.street 125.surfboard 126.table 127.tail
128.tie 129.tile 130.tire 131.toilet 132.towel 133.tower 134.track 135.train 136.tree 137.truck 138.trunk 139.umbrella
140.vase 141.vegetable 142.vehicle 143.wave 144.wheel 145.window 146.windshield 147.wing 148.wire 149.woman 150.zebra.
Given the lexeme, the task is to find semantically relevant lexeme from the predefined entity lexicon.
However, if there is no semantically relevant lexeme in the predefined entity lexicon, please answer 0.None.

In-context Examples

Let's take a few examples.
Question: Given the lexeme "water," find semantically relevant lexeme in the predefined entity lexicon. Answer: *0.None*
Question: Given the lexeme "bus," find semantically relevant lexeme in the predefined entity lexicon. Answer: *142.vehicle*
Question: Given the lexeme "steel," find semantically relevant lexeme in the predefined entity lexicon. Answer: *0.None*
Question: Given the lexeme "vanity," find semantically relevant lexeme in the predefined entity lexicon. Answer: *110.shelf*
Question: Given the lexeme "desktop," find semantically relevant lexeme in the predefined entity lexicon. Answer: *72.laptop*
Question: Given the lexeme "cobble," find semantically relevant lexeme in the predefined entity lexicon. Answer: *104.rock*
Question: Given the lexeme "poles," find semantically relevant lexeme in the predefined entity lexicon. Answer: *99.pole*
Question: Given the lexeme "wastebasket," find semantically relevant lexeme in the predefined entity lexicon. Answer: *6.basket*
Question: Given the lexeme "blue," find semantically relevant lexeme in the predefined entity lexicon. Answer: *0.None*
Question: Given the lexeme "motorcyclist," find semantically relevant lexeme in the predefined entity lexicon. Answer: *98.player*
Question: Given the lexeme "passenger," find semantically relevant lexeme in the predefined entity lexicon. Answer: *91.person*
Question: Given the lexeme "pigeon," find semantically relevant lexeme in the predefined entity lexicon. Answer: *12.bird*
Question: Given the lexeme "grass," find semantically relevant lexeme in the predefined entity lexicon. Answer: *96.plant*
Question: Given the lexeme "surfboards," find semantically relevant lexeme in the predefined entity lexicon. Answer: *125.surfboard*
Question: Given the lexeme "striped shirts," find semantically relevant lexeme in the predefined entity lexicon. Answer: *111.shirt*

Actual Question

Question: Given the lexeme Input, find semantically relevant lexeme in the predefined entity lexicon. Answer:

Table 7. Prompt for alignment of entity class.

Task Description

The predefined predicate lexicon containing 50 lexemes is numbered as follows: 1.above 2.across 3.against 4.along 5.and 6.at
7.attached to 8.behind 9.belonging to 10.between 11.carrying 12.covered in 13.covering 14.eating 15.flying in 16.for 17.from
18.growing on 19.hanging from 20.has 21.holding 22.in 23.in front of 24.laying on 25.looking at 26.lying on 27.made of
28.mounted on 29.near 30.of 31.on 32.on back of 33.over 34.painted on 35.parked on 36.part of 37.playing 38.riding 39.says
40.sitting on 41.standing on 42.to 43.under 44.using 45.walking in 46.walking on 47.watching 48.wearing 49.wears 50.with.
Given the lexeme, the task is to find semantically relevant lexeme from the predefined predicate lexicon.
However, if there is no semantically relevant lexeme in the predefined predicate lexicon, please answer 0.None.

In-context Examples

Let's take a few examples.
Question: Given the lexeme "next to," find semantically relevant lexeme in the predefined predicate lexicon. Answer: *29.near*
Question: Given the lexeme "are parked in," find semantically relevant lexeme in the predefined predicate lexicon. Answer: *35.parked on*
Question: Given the lexeme "waiting," find semantically relevant lexeme in the predefined predicate lexicon. Answer: *0.None*
Question: Given the lexeme "sitting," find semantically relevant lexeme in the predefined predicate lexicon. Answer: *40.sitting on*
Question: Given the lexeme "grazing," find semantically relevant lexeme in the predefined predicate lexicon. Answer: *14.eating*
Question: Given the lexeme "pointing to," find semantically relevant lexeme in the predefined predicate lexicon. Answer: *0.None*
Question: Given the lexeme "lies on," find semantically relevant lexeme in the predefined predicate lexicon. Answer: *24.lying on*
Question: Given the lexeme "sitting underneath," find semantically relevant lexeme in the predefined predicate lexicon. Answer: *43.under*
Question: Given the lexeme "placed next to," find semantically relevant lexeme in the predefined predicate lexicon. Answer: *29.near*
Question: Given the lexeme "looking down at," find semantically relevant lexeme in the predefined predicate lexicon. Answer: *25.looking at*
Question: Given the lexeme "containing," find semantically relevant lexeme in the predefined predicate lexicon. Answer: *0.has*
Question: Given the lexeme "perched on," find semantically relevant lexeme in the predefined predicate lexicon. Answer: *40.sitting on*
Question: Given the lexeme "driving," find semantically relevant lexeme in the predefined predicate lexicon. Answer: *0.None*
Question: Given the lexeme "hangs on," find semantically relevant lexeme in the predefined predicate lexicon. Answer: *19.hanging from*

Actual Question

Question: Given the lexeme Input, find semantically relevant lexeme in the predefined predicate lexicon. Answer:

Table 8. Prompt for alignment of predicate class.

we begin by separating 1,594 entity classes into sub-groups,

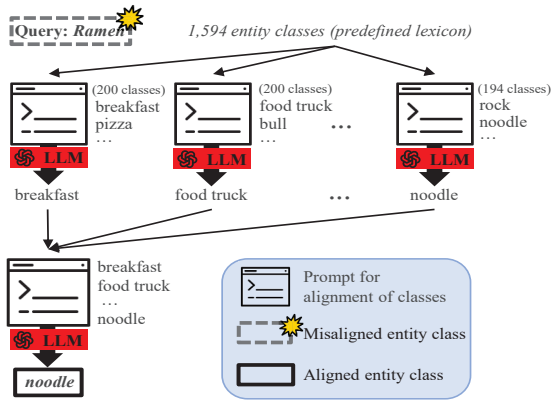


Figure 5. Framework for addressing large predefined lexicons when aligning classes with those of interest.

each of which contains 200 or less than 200 entity classes. It enables us to avoid the maximum prompt length restriction, and list all the entities of sub-groups within the prompt. Then, we can ask the LLM to align the query entity (e.g., Ramen) with entities in the sub-group, yielding semantically relevant entities within each sub-group (e.g., breakfast, food truck, and noodle). Finally, we ask the LLM once more to align a query entity with the intermediate output, allowing for identifying the most semantically relevant entity (e.g., noodle) within the large predefined lexicon. Through this process, we can successfully align a query entity with entity classes in a large predefined lexicon. Aligning predicate classes in a larger predifined lexicon can be done similarly.

A.3. Details of Grounding Methods

To ground unlocalized triplets, we employ two state-of-the-art grounding methods, i.e., SGNLS [57] and VS^3 [54]. Herein, we provide a detailed explanation of each grounding method.

SGNLS. SGNLS employs a pre-trained Faster R-CNN [29] object detector trained on Open Images [14] to ground unlocalized triplets. More precisely, it grounds the subject and object within the unlocalized triplet with image regions that share the same class with the subject/object. It is important to note that the set of 601 entity classes in Open Images does not completely cover the 150 entity classes in Visual Genome [13]. In other words, there are entity classes in Open Images that do not exist in Visual Genome. Therefore, a knowledge base (i.e., WordNet [25]) is used to align as many of Open Images' entity classes as possible with Visual Genome's entity classes. The aligned entity classes of Open Images are then used to compare with the subjects and objects in the unlocalized triplet for grounding. The predicate within the localized subject and object serves as a pseudo label for training the SGG model.

VS^3. VS^3 employs a grounding-based object detector (i.e., GLIP [18]) to ground unlocalized triplets. Specifically, GLIP disentangles the task of object localization, which involves identifying the object's bounding box, and object recognition, which entails recognizing the class associated with that bounding box. Unlike the simultaneous object detection and recognition through Region Proposal Network (RPN) in Faster R-CNN, GLIP initially detects bounding boxes. Then, given the text features corresponding to the entity classes in the target data, it calculates the similarity between these text features [7] and the visual features [6] of bounding boxes. The grounding of the subject and object within an unlocalized triplet is achieved by choosing the bounding boxes with the highest similarity scores. Once subject and object grounding is achieved, the predicate serves as a pseudo-label for training the SGG model.

A.4. Details of Model Training

Here, we provide a detailed explanation for model training after obtaining localized triplets, i.e., $\mathbf{G}_w = \{\mathbf{s}_i, \mathbf{p}_i, \mathbf{o}_i\}_{i=1}^{N_w}$, where $\mathbf{s}_{i,b}$ and $\mathbf{o}_{i,b}$ are obtained from grounding methods. We follow the original training strategy for each method, i.e., SGNLS [57] and VS^3 [54].

SGNLS. SGNLS uses a Transformer-based Uniter model [4] on top of a pre-trained Faster R-CNN [29] object detector to capture contextual information of neighboring objects. Each contextualized representation of the subject and object is input into an entity classifier to generate entity logits. The entity classifier and Uniter model are then trained using cross-entropy loss, with supervision provided by entity labels (i.e., $\mathbf{s}_{i,c}$ and $\mathbf{o}_{i,c}$), respectively. For the predicate, its representation is obtained by feed-forwarding the contextualized representations of the subject and object and is fed into the predicate classifier. Then, the predicate classifier and Untier model are trained using cross-entropy loss with supervision on predicate class $\mathbf{p}_{i,c}$.

VS^3. VS^3 builds an additional predicate classifier on top of a pre-trained GLIP [18] object detector for predicate prediction. Specifically, the concatenation of visual features and spatial information of $\mathbf{s}_i$ and $\mathbf{o}_i$ is fed into MLP to obtain the predicate representation. Based on its representation, the predicate classifier generates the predicate's logit. The predicate classifier and a cross-modal fusion module within GLIP are trained with supervision provided by predicate class $\mathbf{p}_{i,c}$ using cross-entropy loss. When training entities (subject and object), the approach is similar to the class recognition task in GLIP. Specifically, it maximizes the dot product between the text features of entity classes (i.e., $\mathbf{s}_{i,c}$ and $\mathbf{o}_{i,c}$) and their visual features of bounding boxes using binary focal loss [22].

B. Regarding the impact of grounding method on LLM4SGG

In Table 2 of main paper, we observe that applying LLM4SGG to VS^3 [54] (i.e., VS^3+LLM4SGG) results in

greater performance improvement compared to applying it to SGNLS [57] (i.e., SGNLS+LLM4SGG). We provide a detailed explanation regarding the impact of the grounding method on LLM4SGG.

As mentioned in Section A.3, SGNLS includes the process of aligning the 601 entity classes from the Faster R-CNN [29] trained on Open Images [14] with the 150 entity classes in Visual Genome [13]. We find that 34 out of 150 entity classes in Visual Genome are not aligned in the end. In other words, the 601 entity classes in Open Images do not cover these 34 entity classes. As a result, unlocalized triplets containing these 34 entity classes are discarded and not used for training since the image regions do not contain the corresponding 34 classes, and they fail to be grounded. In fact, 100K among the 344K unlocalized triplets obtained through LLM4SGG are discarded, exacerbating the low-density scene graph issue. On the other hand, VS^3 fully utilizes all 344K triplets. As mentioned in Section A.3, VS^3 computes the similarity between text features [7] of entities (subject and object) in the unlocalized triplet and the visual feature [6] of each image region. Then, the image region with the highest score is grounded with that entity. This indicates that the subject and object are always successfully grounded with the image regions having the highest score. Therefore, all 344K triplets are being grounded and used for training, effectively alleviating the low-density scene graph issue.

Taking a further step, in Section 3.4, we use LLM4SGG to align the 601 entity classes in Open Images with 150 entity classes in Visual Genome. However, we observe that even if an LLM is used for alignment, 30 entity classes are still not aligned because 30 entity classes have completely different semantic meanings with the 601 entity classes in Open Images. For example, phone and racket do not overlap with the semantic meaning with 601 entity classes in Open Images. This suggests that the grounding method (i.e., SGNLS) with Faster R-CNN trained on Open Images somewhat limits the effectiveness of LLM4SGG.

C. Experiment Setup

C.1. Dataset

Visual Genome dataset [13] and GQA dataset [11] are widely used in the SGG task. Visual Genome dataset is a benchmark dataset used for evaulating the fully-supervised approach [12, 19, 48, 51]. For test data, each image has 13.8 objects and 6.9 predicates on average.

Recently, GQA dataset has also been used for evaluating the fully-supervised approach [8, 10, 17, 34]. For test data, each image contains 9.3 objects and 4.6 predicates on average.

The predicate distributions of Visual Genome and GQA are plotted in Figure 6.

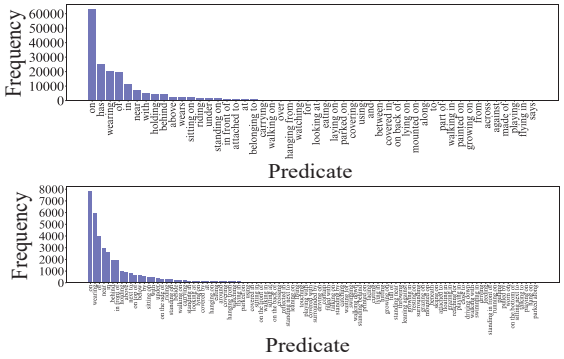


Figure 6. Predicate distribution for Visual Genome (Top) and GQA (Bottom) in test data.

C.2. Evaluation Protocol

In the SGDet task, a predicted triplet is considered as correct if regions corresponding to the subject and object overlap with the ground-truth boxes with an IoU>0.5, while also having the correct subject and object labels. In the procedure of incorporating the correct triplet into R@K and mR@K performance, we initially compute the triplet score by multiplying the subject score, predicate score, and object score for all subject-object pair in an image, followed by sorting them. If the correct triplet falls within the top-K of the sorted triplets, it contributes to R@K and mR@K performance.

C.3. Baselines

For comparison LLM4SGG with baselines, we incorporate the fully-supervised approach [50] and weakly-supervised approach [20, 47, 54, 57] [47].

- **Motif** [50] (Fully-supervised): Based on the analysis of repeated patterns (i.e., motif), this method employs Bi-LSTM to capture the motifs appearing across images.
- **LSWS** [47]: This method utilizes a Graph Neural Network (GNN) applied to triplets extracted from captions to capture the linguistic structure present among triplets, with the aim of improving grounding unlocalized triplets. Furthermore, this method extends its capability by iteratively estimating scores between image regions and text entities.
- **SGNLS** [57]: To ground the unlocalized triplets over image regions, it leverages information from a pre-trained object detector (i.e., Faster R-CNN [29]). Specifically, when the class of a text entity matches a class within a bounding box, the text entity is grounded on that bounding box. Once localized triplets are acquired, a Transformer-based Uniter model [4] is trained based on the contextualized representation of entities under the supervision of localized triplets.
- **[20]**: In the process of grounding unlocalized triplets, this method not only leverages a pre-trained object detec-

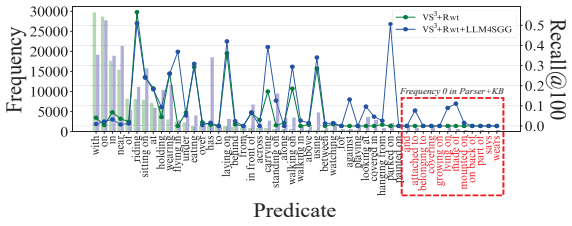


Figure 7. Performance comparison per class when adopting the reweighting method. The red-colored predicates denote the predicates with a frequency of 0 in the conventional approach while LLM4SGG generates all of them with a frequency greater than 0. (Bar: number of predicate instances, Line: Recall@100)

tor [29] for object-aware information but also leverages a pre-trained visual-language model [15] to incorporate interaction-aware information.

- **VS^3** [54]: This method employs a pre-trained object detector (i.e., GLIP [18]) to accomplish more than grounding unlocalized triplets; it also aids in identifying novel entities within these triplets. In contrast to earlier WSSGG works that rely on a Faster R-CNN object detector for grounding, this method capitalizes on the grounding ability of the GLIP that disentangles the tasks of class recognition and localization, leading to a significant enhancement of grounding effectiveness.

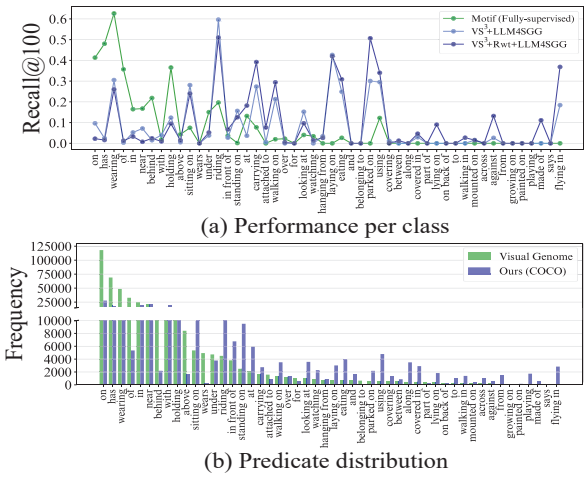
C.4. Implementation Details

In the grounding process in SGNLS [57], we used Faster R-CNN [29] as the object detector, which is pre-trained on Open Images [14]. In the grounding process in VS^3, we used GLIP [18] as the object detector with the Swin-L backbone [23]. Regarding an LLM, we use *gpt-3.5-turbo* in ChatGPT [27]. Note that to further alleviate the long-tailed predicate distribution after Step 3 in our framework, we select the most fine-grained predicate when there are multiple predicates between the same subject-object pair, where the fine-grainedness is determined based on the predicate distribution within the entire set of unlocalized triplets. For more insights regarding the impact of the predicate selection, please refer to Section D.3.

D. Experiment on VG

D.1. Performance Comparison per class with Reweight method

In Figure 7, we show the performance per class of LLM4SGG adopted to VS^3 (i.e., VS^3+LLM4SGG) and a baseline (i.e., VS^3). We observe that the performance of VS^3 + Rwt on 22 fine-grained predicates, which start from the rightmost end of the x-axis *wears* to *between*, drops to nearly zero, although we attempt to enhance the performance of fine-grained predicates with the reweighting method. This is due to the fact that the conventional



(a) Performance per class

(b) Predicate distribution

Figure 8. (a) Performance comparison per class with fully-supervised approach (i.e., Motif, [50]). (b) Predicate class distribution. The x-axis is sorted by the Visual Genome's predicate frequency in descending order.

approach generates unlocalized triplets with a limited number of predicates, and 12 of them even have a frequency of 0. This scarcity of predicates makes it challenging to improve the performance of fine-grained predicates even with reweighting. In contrast, LLM4SGG addresses the semantic over-simplification inherent in the extraction of triplets from captions, which increases the number of fine-grained predicate instances. As a result, when the reweighting method is employed, it effectively boosts the performance of fine-grained predicates. It is worth noting that for some predicates whose frequency is 0 in the conventional approach (i.e., *attached to, lying on, made of, mounted on*), LLM4SGG shows performance improvements, verifying the effectiveness of LLM4SGG.

D.2. Detailed Performance Comparison with fully-supervised approach

In Table 2 of the main paper, we observe that VS^3+LLM4SGG (or VS^3+Rwt+LLM4SGG) achieves higher mR@K performance compared with Motif [50], which is trained using the Visual Genome dataset in a fully-supervised manner, while it shows a lower R@K performance. To delve into this further, we present the performance of each predicate class in Figure 8(a) and the predicate class distribution in Figure 8(b). As shown in Figure 8(a), Motif exhibits a higher performance on coarse-grained predicates (e.g., on, has, of) than on fine-grained predicates (e.g., laying on, parked on). This is due to the fact that its prediction is biased towards coarse-grained predicates, suffering from the long-tailed predicate class distribution of the dataset shown in Figure 8(b). As indicated by the predicate class distribution of Visual Genome's

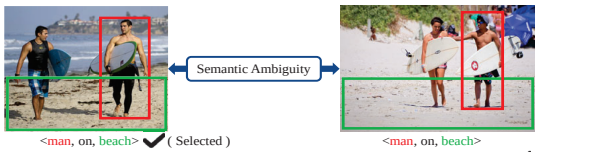


Figure 9. Example of illustrating semantic ambiguity between (a) and (b).

Row	Selection	R@50 / 100	mR@50 / 100	F@50 / 100
	VS^3+LLM4SGG			
(a)	Random	8.15 / 9.55	5.10 / 6.19	6.27 / 7.51
(b)	Coarse-grained	**9.79 / 11.37**	2.52 / 3.03	4.01 / 4.78
(c)	Fine-grained	8.91 / 10.43	**7.11 / 8.18**	**7.91 / 9.17**
	VS^3			
(d)	Random	6.60 / 8.01	2.88 / 3.25	4.01 / 4.62
(e)	Coarse-grained	**6.99 / 8.20**	2.66 / 2.99	3.85 / 4.38
(f)	Fine-grained	6.18 / 7.43	**3.82 / 4.27**	**4.72 / 5.42**

Table 9. Experiment for predicate selection

test set shown in Figure 6, focusing on coarse-grained predicates improves the performance in terms of R@K. Therefore, Motif shows high R@K performance while deteriorating the mR@K performance. On the other hand, as shown in Figure 8(b), the predicate class distribution of triplets generated by LLM4SGG is less severe compared with that of the Visual Genome dataset. As a result, VS^3+LLM4SGG trained with the dataset generated by LLM4SGG shows performance improvement on fine-grained predicates but relatively inferior performance on coarse-grained predicates. Therefore, VS^3+LLM4SGG outperforms Motif in terms of mR@K while showing inferior performance on R@K. Furthermore, applying the reweighting strategy (VS^3+Rwt+LLM4SGG) further enhances the performance of fine-grained predicates, leading to an improvement in mR@K. The performance improvement in terms of mR@K compared to the fully-supervised approach, which indicates the capability of constructing richer scene graphs, demonstrates the effectiveness of LLM4SGG.

D.3. Experiment for Predicate Selection

Regarding the implementation details, to further alleviate the long-tailed predicate distribution after Step 3 in our framework, we select the most fine-grained predicate when there are multiple predicates between the same subject-object pair, where the fine-grainedness is determined based on the predicate distribution within the entire set of unlocalized triplets. In Table 9, we further conduct experiments under three different scenarios to understand the effect of predicate selection: random predicate selection, coarse-grained predicate selection, and fine-grained predicate selection. We have the following observations: **1)** LLM4SGG with the random selection (row (a)) and the selection of coarse-grained (row (b)) and fine-grained predicates (row (c)) consistently outperform the baseline with the selection of each case (row (d),(e),(f)), respectively, which verifies the effectiveness of LLM4SGG. **2)** LLM4SGG with a selection of coarse-grained predicates (row (b)) severely deteriorates the mR@K performance while increasing the R@K performance compared to the random selection (row (a)). While R@K performance is improved by increasing the instances of coarse-grained predicates, the fine-grained predicates in an image are not effectively utilized when combined with coarse-grained predicates, resulting in a decrease in the mR@K performance. In contrast, selecting the fine-grained predicates (row (c)) significantly increases the performance of mR@K. **3)** LLM4SGG with a selection of the fine-grained predicates (row (c)) yields higher R@K and mR@K compared to random selection (row (a)). Regarding the improvement of R@K performance, we attribute it to the effect of alleviating the semantic ambiguity [51]. For example, Figure 9(a) and (b) illustrate the two cases where unlocalized triplets parsed from several captions exhibit predicates on and walking on between the same subject man and object beach. If we randomly select the predicates, the model may learn on in one image and walking on in another image even though they are associated with the same subject and object, making the SGG model confused due to the similar visual features with different predicates. For this reason, selecting walking on for both images helps mitigate semantic ambiguity, resulting in enhancing the performance.

Method	zR@50	zR@100
Motif (Fully-supervised)	0.31	0.60
VS^3	1.16	1.46
VS^3+LLM4SGG	**2.20**	**3.02**

Table 10. Performance comparison for exploration of training space.

D.4. Experiment for Exploration of Training Space

In Table 10, we show the results of zero-shot Recall@K (zR@K), as introduced by [35], to explore the training space of triplets extracted from captions. It is important to note that the zR@K metric used in the fully-supervised approach evaluates how well the model predicts ⟨subject, predicate, object⟩ triplet sets that have never been observed in the training data of Visual Genome dataset [13]. Therefore, while triplets extracted from the captions in COCO dataset may accidentally overlap with these zero-shot triplets, we employ this metric to understand how much our proposed approach broadens the training space. When we compare VS^3, which learns from triplets extracted from captions through the conventional approach, to Motif, a fully-supervised approach trained on Visual Genome dataset, we observe that VS^3 achieves a higher zR@K, implying that captions contain a broader range of diverse com-

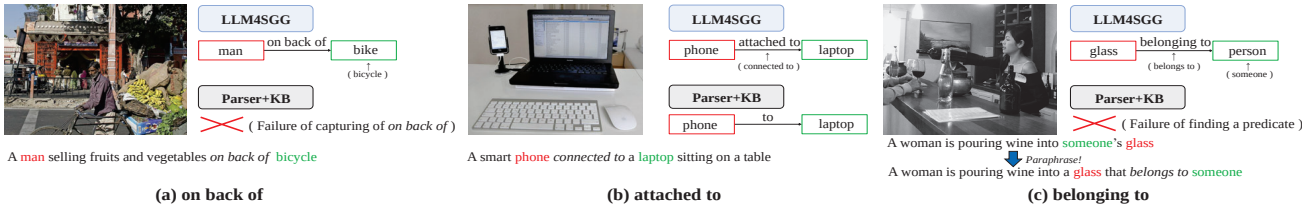


Figure 10. Case studies on extracting fine-grained predicates with a frequency of 0 in the conventional approach (i.e., Parser+KB). (a): LLM4SGG extracts the fine-grained predicate found in the caption. (b): LLM4SGG extracts the fine-grained predicate while aligning it with a predicate in the target data. (c): In LLM4SGG, paraphrasing the caption via LLM aids in identifying a fine-grained predicate, imparting a more specific meaning to it.

Method	R@50 / 100	mR@50 /100	F@50 / 100
VS^3	6.60 / 8.01	2.88 / 3.25	4.01 / 4.62
VS^3+LLM4SGG$_{comb}$	8.66 / 10.20	6.28 / 7.06	7.28 / 8.34
VS^3+LLM4SGG	**8.91 / 10.43**	**7.11 / 8.18**	**7.91 / 9.17**

Table 11. Experiment for new prompt.

positional relationships compared to the Visual Genome dataset. On the other hand, VS^3 with LLM4SGG (i.e., VS^3+LLM4SGG) significantly improves the zR@K performance compared to VS^3. This suggests that triplets generated through LLM4SGG are proficient at capturing the compositional relationships found in captions, thereby expanding the training space of triplets. This expansion is achieved by addressing the semantic over-simplification issue, leading to the creation of a more varied set of predicates, and the low-density scene graph, resulting in a wider range of compositional triplets. We argue that the use of the zR@K metric demonstrates the effectiveness of LLM4SGG in terms of expanding the training space.

D.5. Experiment for New Prompt Design

In Table 11, we conduct an experiment with a new prompt design. In fact, the prompt for extracting triplets from captions in Section 3.3 and the alignment of entity/predicate classes with target data in Section 3.4 can be combined into one. More precisely, we instruct the LLM to follow the four steps for triplet extraction from paraphrased caption: Step 1) Paraphrase the caption, Step 2) Extract meaningful triplets from the paraphrased caption obtained in Step 1, Step 3) Find semantically relevant lexeme in the predefined entity lexicon for the subject and object from the triplets obtained in Step 2, where the entity lexicon is enumerated in Table 7. Step 4) Find semantically relevant lexeme in the predefined predicate lexicon for the predicate from the triplets obtained in Step 3, where the predicate lexicon is enumerated in Table 8. Following the four steps, we include stepwise results in the in-context examples for in-context few-shot learning, and insert the caption from which we want to extract triplets in the actual question. As shown in Table 11, LLM4SGG with the combined prompt (i.e., VS^3+LLM4SGG$_{comb}$) outperforms the baseline, implying that even though prompt design can be diverse, it consistently verifies the efficacy of the LLM-based triplet formation process. On the other hand, VS^3+LLM4SGG$_{comb}$ shows inferior performance compared to VS^3+LLM4SGG. This is due to a practical reason incurred by the length limit in the prompt of GPT-3.5, which allows only up to 4096 tokens. Specifically, VS^3+LLM4SGG$_{comb}$ integrates the instructions of all four steps in the task description, including the definition of entity/predicate lexicons, and in-context examples following four steps within a single prompt, which leads to a longer length while the maximum length is constrained. For this reason, we could only accommodate four in-context examples. In contrast, VS^3+LLM4SGG divides the four steps into two chains (Section 3.3 and Section 3.4), allowing more in-context examples in each chain. Specifically, VS^3+LLM4SGG contains six in-context examples in Chain-1 (i.e., triplet extraction) and fourteen in-context examples in Chain-2 (i.e., alignment of entity/predicate classes). The increased number of in-context examples equips an LLM to adapt more effectively to the provided few-shot task [41], ultimately enhancing its adaptability for the task of triplet extraction task and alignment of entity/predicate classes. In summary, under the practical length limit of GPT-3.5, we observe that an approach that divides the original four steps into two chains is more effective in extracting triplets aligned with entity/predicate of interest than an approach that combines them into a single prompt.

D.6. Case Studies on Extracting Fine-grained Predicates

In Figure 10, we present case studies on predicates in which the conventional approach (i.e., Parser+KB) eventually ends up in a frequency of 0, while LLM4SGG does not. In Figure 10(a), despite the presence of the fine-grained predicate on back of in the caption, the conventional approach fails to extract it since the conventional approach relies on a heuristic rule-based parser [43] that lacks the capability of understanding the context of caption and extracting the predicate on back of at once. On the other hand, LLM4SGG

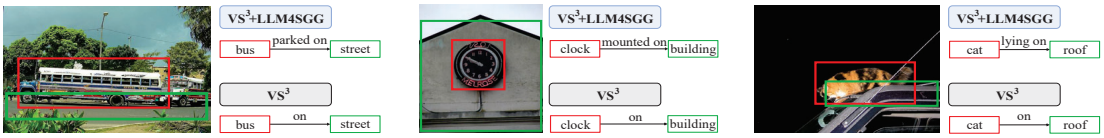


Figure 11. Qualitative results on Visual Genome dataset. (a) A predicate *parked on* rarely appears in the conventional approach. (b), (c) Predicates *mounted on* and *lying on* never appear in the conventional approach.

successfully extracts the predicate on back of through a comprehensive understanding of the caption's context. In Figure 10(b), we observe that the conventional approach, which suffers from semantic over-simplification, extracts the coarse-grained predicate to instead of connected to, whereas LLM4SGG extracts the fined-grained predicate connected to within the caption. Subsequently, by aligning connected to with semantically relevant lexeme, attached to, in the target data through LLM's semantic reasoning ability, it eventually generates a fine-grained predicate attached to. This demonstrates the effectiveness of LLM-based triplet extraction in addition to LLM-based alignment, leading to capturing the fine-grained predicates. Interestingly, in Figure 10(c), we observe that the paraphrasing step (Section 3.3) in LLM4SGG aids in extracting fine-grained predicates. Specifically, paraphrasing the caption conveys a more specific meaning, i.e., from *someone's glass* to *glass that belongs to someone*, thus enabling the extraction of the fine-grained predicate belonging to, which the conventional approach cannot achieve. Through these case studies, we demonstrate the effectiveness of extracting the fine-grained predicates in LLM4SGG.

D.7. Qualitative Results

To further verify the effectiveness of LLM4SGG on Visual Genome dataset [13], in Figure 11 , we showcase qualitative results from the test data, comparing the baseline (i.e., VS^3 [54]) with LLM4SGG applied to VS^3 (i.e., VS^3+LLM4SGG) . In Figure 11(a), we observe that LLM4SGG accurately predicts a fine-grained predicate parked on between subject bus and object street, which rarely appears in the training data using the conventional approach. In contrast, the baseline (i.e., VS^3) following the conventional approach predicts a coarse-grained predicate on due to the semantic over-simplification, incurring the long-tailed predicate distribution. Furthermore, in Figure 11(b), we observe that LLM4SGG makes a correct prediction on the predicate whose frequency is 0 in the conventional approach (i.e., mounted on). On the other hand, the baseline predicts a coarse-grained predicate on and never predicts mounted on since it has not been observed during training. Interestingly, while in Figure 11(c) LLM4SGG made an incorrect prediction by predicting lying on instead of on, we argue that lying on is a more realistic answer that provides a richer context. However, as lying on is never observed while training the baseline (i.e., VS^3), it can never be predicted. Through the qualitative analysis, we again demonstrate the effectiveness of alleviating the long-tailed problem in WSSGG.

E. Experiment on GQA

E.1. Training and Evaluation

Training. To train a SGG model for evaluation on the GQA dataset [11], LLM4SGG requires three modifications, encompassing the change of 1) predefined entity, 2) predicate lexicon, and 3) in-context examples in Section 3.4, while maintaining the triplet extraction process in Section 3.3. More precisely, in the predefined entity and predicate lexicons, we replace the entity and predicate classes originally associated with Visual Genome dataset [13] with entity and predicate classes in the GQA dataset, respectively. For in-context examples, we substitute the examples that are initially relevant to Visaul Genome dataset with examples related to GQA dataset. After obtaining unlocalized triplets composed of entity/predicate classes in the GQA dataset, the process of grounding them remains unchanged from the grounding method used in Visual Genome dataset. Please refer to the details of the grounding process in Section A.3.

Evaluation. To evaluate a trained model on the GQA dataset, only a single modification is needed in the architecture of VS^3 [54]. Specifically, we change the output entity class of bounding boxes from the entity classes of the Visual Genome dataset to those of the GQA dataset. For details of VS^3 regarding the determination of entity classes for bounding boxes, the target data's entity classes are listed in text format, e.g., airplane. animal. arm. ... . Then, a text encoder [7] encodes the enumerated text list to obtain the text features for each entity. The similarity between these text features of entity classes and visual features [6] of the bounding boxes is computed. In perspective of the bounding box, the text entity with the highest similarity is chosen as the class assigned to the bounding box. In the procedure of determining the entity classes for bounding boxes, we simply list the entity classes from the GQA dataset instead of Visual Genome's entity classes. This ensures that entity

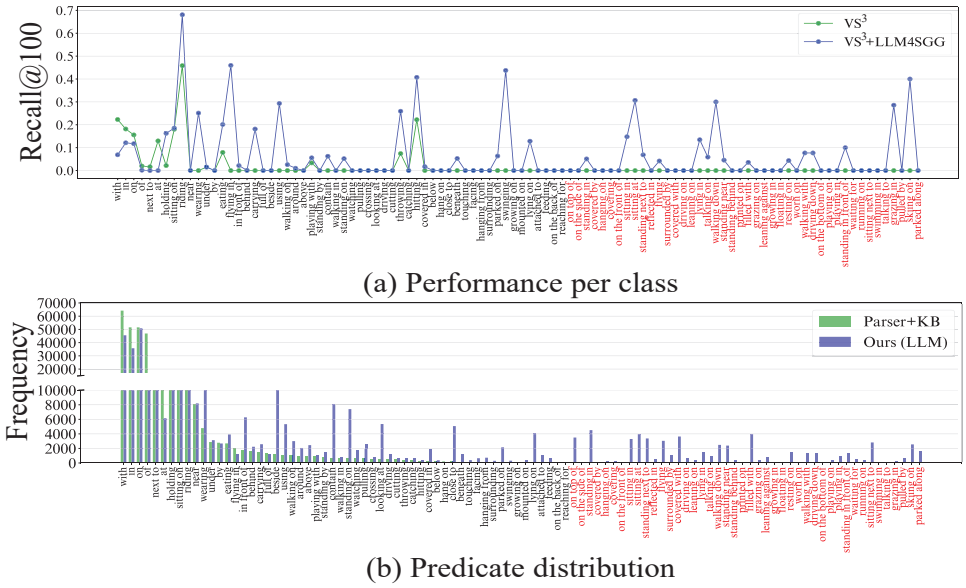


Figure 12. (a) Performance comparison per class, (b) Predicate distribution on GQA dataset. The red-colored predicates indicate that the frequency of predicate instances generated by the conventional approach is 0, while there are no predicates with a frequency of 0 in LLM4SGG.

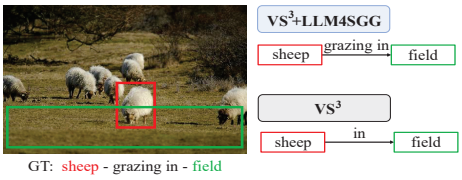
classes assigned to the bounding boxes align with the entity classes in the GQA dataset. Subsequently, we proceed with a standard evaluation protocol used in Visual Genome dataset.

E.2. Performance Comparison per class

In Figure 12(a), we show a performance comparison per class on the GQA dataset [11]. For baseline (i.e., VS^3 [54]), we observe that the performance for most predicates, except for coarse-grained predicates, is nearly zero. It is attributed to the semantic over-simplification inherent in the conventional approach, leading to a long-tailed predicate class distribution as shown in Figure 12(b). The long-tailed problem is severely exacerbated in the GQA dataset [11] due to its inclusion of more complicated predicates (e.g., standing next to), as well as a greater variety of fine-grained predicates, such as talking on and driving down, which are challenging to extract from captions using heuristic rule-based parser [43]. In fact, 44 out of 100 predicates have a frequency of 0, meaning that they are never predicted. On the other hand, as shown in Figure 12(b), LLM4SGG effectively addresses the long-tailed problem by alleviating the semantic over-simplification and low-density scene graph issues, increasing the instances that belong to the fine-grained predicates instances so that there are no predicates with a frequency of 0. As a result, LLM4SGG significantly enhances the performance of fine-grained predicates, thereby improving the performance of mR@K. This demonstrates the effectiveness of LLM4SGG with the more challenging GQA dataset.

E.3. Qualitative Results

To further demonstrate the effectiveness of LLM4SGG on a more challenging dataset, GQA [11], we present qualitative results comparing the baseline (i.e., VS^3) with LLM4SGG applied to VS^3 (i.e., VS^3+LLM4SGG) in Figure 13. For Figure 13(a), (b), and (c), we showcase examples from the test data, where predicates have a frequency of 0 in the training data when triplets are generated using the conventional approach, i.e., grazing in, skiing on, and standing in front of. In Figure 13(a) and (b), we observe that the baseline predicts the coarse-grained predicate in, which is the second most frequent predicate, as shown in Figure 12(b). The prediction of coarse-grained predicate is attributed to the semantic over-simplification issue of the conventional approach, leading to a long-tailed problem. On the other hand, LLM4SGG correctly predicts the fine-grained predicates grazing in and skiing on by effectively alleviating the semantic over-simplification issue. In Figure 13(c), we encounter a complicated predicate, standing in front of, between the subject woman and object door. Such predicates are intricate to extract from captions unless they are comprehensively understood and extracted at once. LLM4SGG, however, adeptly extracts the fine-grained predicate standing in front of by understanding the entire context of the caption via LLM. LLM4SGG makes it possible to learn the

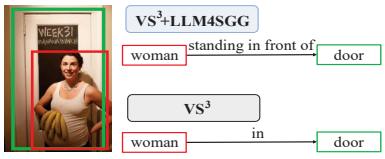


GT: sheep - grazing in - field

(a) Predicate of Frequency 0 (*grazing in*)

GT: person - skiing on - mountain

(b) Predicate of Frequency 0 (*skiing on*)



GT: woman - standing in front of - door

(c) Predicate of Frequency 0 (*standing in front of*)

Figure 13. Qualitative results on GQA dataset. (a), (b), (c): predicates *grazing in*, *skiing on*, and *standing in front of* never appear while training the baseline VS^3.

predicate standing in front of, resulting in a correct prediction. These qualitative results on the more challenging GQA dataset further verify the effectiveness of LLM4SGG.

F. Replacing LLM with Smaller Language Models

Note that a potential limitation of LLM4SGG is that it relies on a proprietary black box LLM, i.e., GPT-3.5 (135B), to extract triplets from captions and align the triplets with entity/predicate classes of interest, which is costly to use. This raises a question: can we replace the black box LLM with an open-source LLM of a smaller size? To explore this question, we employ LLaMA2-7B [38] as a smaller language model and use it to perform Chain-1 (Section 3.3) and Chain-2 (Section 3.4) with the same prompts shown in Section A.1.

F.1. Quantitative Results

In Table 12, we conduct an experiment on Visual Genome dataset for inference to quantitatively measure how well LLM4SGG works with a smaller language model (i.e., LLaMA2-7B). We observe that LLM4SGG using LLaMA2 (i.e., LLM4SGG-LLaMA 2) outperforms the baseline (i.e., VS^3), especially in terms of mR@K. This implies that LLM4SGG-LLaMA2 effectively alleviates the semantic over-simplification issue. Regarding the mitigation of the low-density scene graph issue, we observe that LLM4SGG-LLaMA2 increases the number of triplets from 154K to 228K, leading to performance improvements in terms of both R@K and mR@K. In summary, LLM4SGG is still superior to the baseline when replacing the LLM with a smaller language model. On the other hand, we observe that LLM4SGG-LLaMA2 underperforms LLM4SGG-GPT-3.5. We attribute this to the fact that compared to LLM4SGG-GPT-3.5, LLM4SGG-LLaMA2 discards more triplets when aligning classes in triplets (i.e., Chain-2). In fact, LLM4SGG with LLaMA2 discards **81.9**% of the triplets obtained after Step 2 (1.25M[4]→288K) during the process of aligning classes in triplets (Chain-2), whereas LLM4SGG-GPT-3.5 only discards **72.6**% of the triplets obtained after Step 2 (1.25M→344K). This discrepancy is mainly due to the disparity in semantic reasoning ability between LLaMA2 and GPT-3.5, attributed to the significance difference in the model size (i.e., 7B vs. 175B).

Method	# Triplet	R@50 / 100	mR@50 / 100	F@50 / 100
VS^3	154K	6.60 / 8.01	2.88 / 3.25	4.01 / 4.62
+LLM4SGG-LLaMA2 (7B)	228K	8.26 / 9.91	5.90 / 6.72	6.88 / 8.01
+LLM4SGG-GPT-3.5 (175B)	344K	**8.91 / 10.43**	**7.11 / 8.18**	**7.91 / 9.17**

Table 12. Performance comparison with the utilization of smaller language model.

Nevertheless, based on the result of LLM4SGG-LLaMA2, we argue that GPT-3.5 in LLM4SGG can be replaced with a smaller language model. Moreover, we expect that using a even larger language model, i.e., LLaMA2-70B [38], would further enhance the performance.

F.2. Qualitative Results

In Figure 14, we show qualitative results related to triplets extracted from captions, where we replace GPT-3.5 with a smaller language model (i.e., LLaMA2-7B). In Figure 14(a), we observe that the utilization of LLaMA2 alleviates the semantic over-simplification. Specifically, LLM4SGG-LLaMA2 extracts fine-grained predicates laying on and standing on within captions while the conventional approach fails to extract it or extract coarse-grained predicate on. In Figure 14(b), we observe that even if LLaMA2 is capable of extracting fine-grained predicates (i.e., walking down, sitting around), it fails to align them with predicate classes of interest, resulting in discarding triplets. On the other hand, GPT-3.5 successfully aligns them with relevant predicates of interest, such as walking on and sitting on. The failure of LLaMA's alignment stems from its limited semantic reasoning ability compared to GPT-3.5. As a result, LLM4SGG-LLaMA2 yields a smaller number of triplets compared with LLM4SGG-GPT-3.5 (228K vs. 344K) so that it shows inferior performance compared to LLM4SGG-GPT-3.5. Moreover, in Figure 14(c), we observe that LLaMA2 occasionally aligns

[4] After Chain-1, 1.25M triplets are extracted.

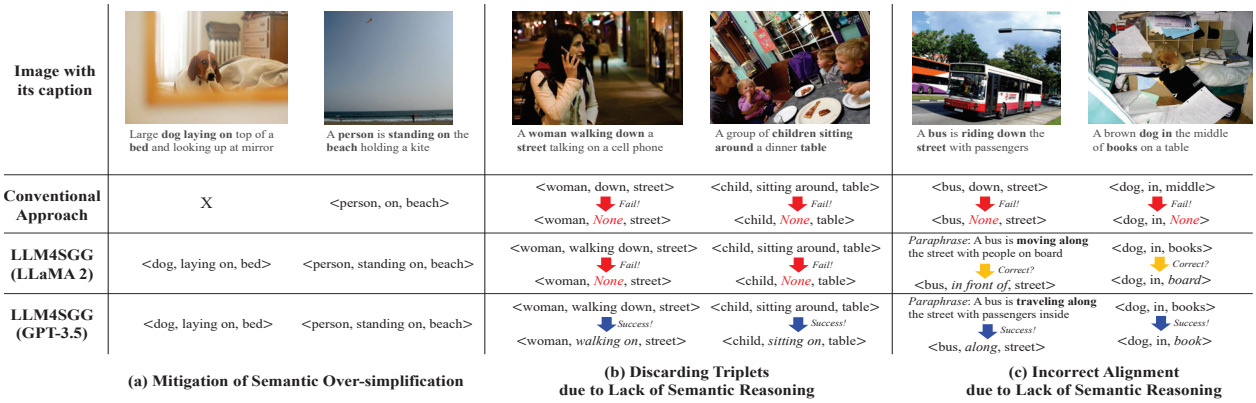


Figure 14. Qualitative Results for utilization of a smaller language model (i.e., LLaMA2-7B). Red arrow: Failure of alignment, Blue arrow: Success of alignment, Orange arrow: Incorrect alignment.

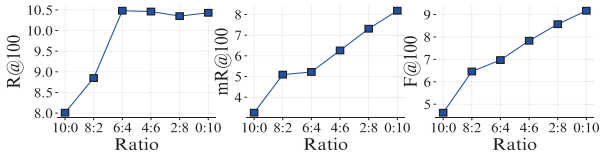


Figure 15. Quantitative study for quality of triplets generated by LLM4SGG. Ratio denotes the ratio of triplets' combination between the conventional approach (front number) and LLM4SGG (rear number).

classes with irrelevant classes of interest. More precisely, LLaMA2 aligns moving along and books with in front of and board, respectively. In contrast, GPT-3.5 aligns them with somewhat semantically relevant classes, such as along and book, respectively. For LLM4SGG-LLaMA2, this alignment with semantically irrelevant classes would also incur performance deterioration, as shown in Table 12. However, as discussed in Section F.1, we expect that these issues could be solved by replacing it with a larger model, i.e., LLaMA2-70B.

G. Evaluating the Quality of Triplets Generated by LLM4SGG

In Figure 15, we quantitatively evaluate the quality of triplets generated by LLM4SGG. To this end, we combine triplets made by the conventional approach (i.e., Parser+KB) and triplets generated by LLM4SGG with different ratios. Specifically, we gradually increase the ratio of triplets generated by LLM4SGG while decreasing the ratio of triplets made by the conventional approach. For example, in an 8:2 ratio, we randomly select 80% of the total triplets made by the conventional approach and 20% of the total triplets generated by LLM4SGG. As shown in Figure 15, as we increase the ratio of LLM4SGG's triplets, we observe a gradual improvement in mR@K and F@K performance.

Steps	Num. Output/Input tokens per image	Cost per image
(Step 2-1) Triplet Extraction of Original Caption	0.16K / 0.52K	$0.00050
(Step 2-2) Triplet Extraction of Paraphrased Caption	0.48K / 0.89K	$0.00117
(Step 3) Alignment of object classes	0.11K / 1.18K	$0.00076
(Step 3) Alignment of predicate classes	0.11K / 0.82K	$0.00058

Table 13. Num. tokens and cost per image containing 5 captions.

This implies that LLM4SGG's triplets contain a higher number of fine-grained predicates and exhibit superior quality. Furthermore, beyond a 0.4 ratio of LLM4SGG (i.e., 6:4 ratio), the R@K performance remains relatively consistent. It is worthwhile to note that considering that triplets made by the conventional approach predominantly consist of coarse-grained predicates, reducing their proportion would generally result in a drop in R@K performance due to the reduction in the number of coarse-grained predicates. Nevertheless, the fact that R@K performance remains consistent after 0.4 ratio indicates that LLM4SGG's triplets contain high-quality coarse-grained predicates to some extent, surpassing the quality of coarse-grained triplets made by the conventional approach. Through the above results, we demonstrate the effectiveness of LLM4SGG in generating high-quality triplets.

H. Cost for Utilization of a Large Language Model

The cost of using ChatGPT for augmenting the data is summarized in Table 13. Considering that the input tokens and output tokens cost $0.0005 and $0.0015 per 1K tokens, the cost per image for Step 2-1 is computed by (520/1K) $\times$ 0.0005 + (160/1K) $\times$ 0.0015, which is similarly computed in other steps.