DiscoSG: Towards Discourse-Level Text Scene Graph Parsing through Iterative Graph Refinement

Shaoqing Lin[1], **Chong Teng**[1†], **Fei Li**[1], **Donghong Ji**[1],
Lizhen Qu[2], **Zhuang Li**[3‡]
[1]Key Laboratory of Aerospace Information Security and Trusted Computing,
Ministry of Education, School of Cyber Science and Engineering, Wuhan University
[2]Faculty of Information Technology, Monash University
[3]School of Computing Technologies, RMIT University
{sqlinn,tengchong,lifei_csnlp,dhji}@whu.edu.cn
lizhen.qu@monash.edu zhuang.li@rmit.edu.au

Abstract

Vision-Language Models (VLMs) generate discourse-level, multi-sentence visual descriptions, challenging text scene graph parsers built for single-sentence caption-to-graph mapping. Current approaches typically merge sentence-level parsing outputs for discourse input, often missing phenomena like cross-sentence coreference, resulting in fragmented graphs and degraded downstream VLM task performance. We introduce a new task, **Disco**urse-level text **S**cene **G**raph parsing (**DiscoSG**), and release **DiscoSG-DS**, a dataset of 400 expert-annotated and 8,430 synthesised multi-sentence caption-graph pairs. Each caption averages 9 sentences, and each graph contains at least 3× more triples than those in existing datasets.

Fine-tuning GPT-4o on DiscoSG-DS yields over 40% higher SPICE metric than the best sentence-merging baseline. However, its high inference cost and licensing restrict open-source use. Smaller fine-tuned open-source models (e.g., Flan-T5) perform well on simpler graphs yet degrade on denser, more complex graphs. To bridge this gap, we introduce **DiscoSG-Refiner**, a lightweight open-source parser that drafts a seed graph and iteratively refines it with a novel learned graph-editing model, achieving 30% higher SPICE than the baseline while delivering 86× faster inference than GPT-4o. It generalises from simple to dense graphs, thereby consistently improving downstream VLM tasks, including discourse-level caption evaluation and hallucination detection, outperforming alternative open-source parsers. Code and data are available at https://github.com/ShaoqLin/DiscoSG.

1 Introduction

Text scene graph parsing converts visual descriptions into graphs of entities and their relations, supporting tasks like image captioning evaluation (Dong et al., 2024), hallucination detection (Yu et al., 2024), and image retrieval (Johnson et al., 2015). Traditional methods target single-sentence captions, converting dependency parses into graphs or fine-tuning Pre-trained Language Models (PLMs) on sentence-level pairs. As Vision-Language Models (VLMs) now generate detailed discourse-level, multi-sentence captions with complex inter-sentential dependencies (Cheng et al., 2025), these methods are increasingly inadequate.

Processing discourse-level text introduces several critical challenges for current sentence-level parsers: **First**, cross-sentence coreference resolu-

Image.	
Caption.	A group of people are seen walking on a concrete pier towards a ferry terminal . . . In the distance, tall buildings loom, indicating that the location is near a city . . . (*details omitted for brevity*)
Init.	(people, walk on, pier), (people, walk towards, ferry terminal), (people, move towards, destination), (pier, is, concrete), (buildings, is, tall)
Deletion.	(people, walk on, pier), (people, walk towards, ferry terminal), ~~**(people, walk towards, destination)**~~, (pier, is, concrete), (buildings, is, tall)
Insertion.	(people, walk on, pier), (people, walk towards, ferry terminal), ~~**(people, walk towards, destination)**~~, (pier, is, concrete), (buildings, is, tall), **(people, is, group of)**
Refined.	(people, walk on, pier), (people, walk towards, ferry terminal), (pier, is, concrete), (buildings, is, tall), (people, is, group of)

Table 1: Illustration of the iterative scene graph refinement process in the DiscoSG-Refiner framework.

[†]Corresponding author.
[‡]Senior author.

Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing, pages 7849–7874
November 4-9, 2025

tion requires correctly linking entities across sentences (e.g., "a woman" later as "she"). **Second**, capturing long-range relations between entities in different sentences. **Third**, inferring implicit relationships not stated in any single sentence (e.g., `(cat, near, window)` from "The cat is on the mat" and "The mat is under the window"). **Finally**, ensuring graph coherence by producing a globally consistent and complete scene representation. However, existing methods (Dong et al., 2024) typically resort to heuristically merging sentence-level parsing outputs, resulting in semantically inconsistent graphs that ignore long-range dependencies.

To address these issues, we define **Disco**urse-level text **S**cene **G**raph parsing (**DiscoSG**), which converts multi-sentence descriptions into scene graphs. We release **DiscoSG-DS**, a dataset of 400 expert-annotated and 8,430 synthesised caption-graph pairs. Each caption averages 9 sentences, and each graph contains at least 3× more subject-predicate-object triples than prior datasets (Li et al., 2023a; Krishna et al., 2017; Yang et al., 2025). Our annotation guidelines ensure graphs capture cross-sentence coreference, long-range dependencies, implicit relations, and global consistency.

However, standard approaches, such as end-to-end fine-tuning of PLMs (Li et al., 2023a) on caption-graph pairs, face challenges with DiscoSG-DS. Fine-tuning large PLMs (e.g., GPT-4o[1]) yields **over 40%** higher SPICE (Anderson et al., 2016) than the strongest sentence-merging baseline. However, it is impractical for open-source use due to slow inference (about **30s per query**), approximately **$25 to $150** per run in API cost on VLM benchmarks (Dong et al., 2024), and restrictive licensing. Few-shot prompting of large PLMs (Yang et al., 2025) also underperforms their fine-tuned counterparts on this discourse task because it lacks task-specific adaptation. Naive fine-tuning of smaller PLMs (e.g., Flan-T5 (Chung et al., 2024)) performs comparably to GPT-4o on the simple-graph test set, yet degrades on the dense-graph set by at least **15%** SPICE, and falls significantly behind the sentence-merging baseline on downstream VLM tasks that involve a broad range of graph complexities, including evaluation of **VLM-generated captions and hallucination detection**.

To overcome the limitations, we propose **DiscoSG-Refiner**, a lightweight iterative graph-refinement framework (Table 1). Starting from a base graph merged from sentence-level parses, an encoder-decoder PLM "Programmer" proposes edits to simplify dense graph construction. We *disentangle* deletion and insertion so the decoder generates shorter edit sequences, which improves editing performance: the encoder deletes erroneous triples, and the decoder inserts triples to improve semantic coverage. Trained on DiscoSG-ED, a synthetic corpus of edit operations derived from DiscoSG-DS, the Programmer learns to exploit the inter-sentence context and task knowledge in DiscoSG-DS. Built from two Flan-T5-base models totalling **0.5B** parameters, DiscoSG-Refiner **balances generalisation, performance, and cost**. It achieves **30%** higher SPICE than the sentence-merging baseline on both simple and dense graph test sets, **without degradation** across graph densities; delivers **86**× faster inference than GPT-4o; and runs on a single GPU at substantially lower cost. On downstream VLM tasks, it consistently improves results, outperforming the strongest sentence-merging parser and end-to-end Flan-T5 parsers.

Due to the lack of resources for discourse-level hallucination detection in VLMs, we introduce D-FOIL, a benchmark inspired by FOIL (Shekhar et al., 2017), originally designed for sentence-level detection. D-FOIL comprises **200** pairs of hallucinated and human-corrected **multi-sentence** captions, with corrections made against a reference.

Overall, our main contributions are: **(I)** defining DiscoSG, a novel task for discourse-level text scene graph parsing from multi-sentence visual descriptions; **(II)** introducing DiscoSG-DS, a dataset of 400 expert-annotated and 8,430 synthesised examples for discourse-level parsing; **(III)** proposing DiscoSG-Refiner, a lightweight iterative framework with a novel PLM-based "Programmer" trained on synthesised edits (DiscoSG-ED) derived from DiscoSG-DS; **(IV)** demonstrating that our 0.5B open-source model outperforms strongest sentence-merging baselines by 30%, and is more resource-efficient than fine-tuned GPT-4o; and **(V)** establishing D-FOIL, a new benchmark for discourse-level hallucination detection.

2 DiscoSG-DS: Discourse-Level Scene Graph Dataset

To study DiscoSG, we introduce the DiscoSG-DS dataset, which consists of 400 manually annotated gold-standard examples and 8,430 synthesised instances. Each instance aligns a discourse-level

[1] https://openai.com/index/hello-gpt-4o/

image description $\boldsymbol{x} = (x_1, \ldots, x_n)$ with a corresponding scene graph $\boldsymbol{y}$. Each description $\boldsymbol{x}$ contains n sentences that are contextually dependent, requiring parsers to model discourse-level semantics. Each graph ($\boldsymbol{y}$) captures the semantics of $\boldsymbol{x}$ through a set of structured triples representing object relationships (e_{sub}, r, e_{obj}) (such as `(man, wear, hat)`) and attributes (e, a, v) (like `(hat, has_attribute, red)`).

2.1 Source Data Selection

Descriptions ($\boldsymbol{x}$) were sourced from the SharedGPT4V dataset (Chen et al., 2024), containing 1.2 million captions generated by GPT-4 for images. These highly descriptive captions span multiple sentences, making them suitable for discourse-level analysis. We selected 40,000 diverse image-description pairs using diversity sampling based on TF-IDF text embeddings, adapting methods from Zhuo et al. (2023); Li et al. (2023b). The selected descriptions were then annotated either manually or via synthesis to form the final DiscoSG-DS dataset.

2.2 Manual Annotation Pipeline

To construct high-quality ground truth data for DiscoSG, we manually annotated 400 descriptions randomly selected from the pool of 40,000, using a human-in-the-loop active learning framework.

Annotator Setup and Annotation Principles. Annotation was conducted by two expert annotators: a senior researcher with over three years of relevant task experience and a trained postgraduate student in computer science. To calibrate the postgraduate annotator, we conducted a phased training process in which the student annotated 40 examples across four batches under the postdoc's supervision. By the final batch, the annotators had achieved 90% inter-annotator agreement.

Text descriptions were first annotated using the FACTUAL-MR intermediate representation (Li et al., 2023a), which reduces annotation ambiguity and enforces a standardised structure, and were then converted into formal scene graphs. The annotation focused on resolving cross-sentence coreference, correcting quantifier inconsistencies, capturing missing long-range relational dependencies, and identifying overlooked implicit relations and semantic contradictions. To ensure the resulting scene graphs accurately reflected the visual content, the protocol also required annotators to correct factual inconsistencies or hallucinations found in the source descriptions ($\boldsymbol{x}$) by verifying against the corresponding images. Detailed annotation guidelines are provided in Section B.2.

Initial Set Creation. We initiated the human-in-the-loop process by generating draft scene graphs for 102 image descriptions, created through heuristic merging of sentence-level parses from FACTUAL-T5 (Li et al., 2023a). Each draft underwent two-stage refinement: the postgraduate annotator first edited the graph, followed by review and finalisation by the postdoctoral researcher, with consensus required for acceptance. From this curated set, 40 examples were reserved as a fixed validation set (D_{val}), and the remaining 62 formed the seed training set (D_{seed}) for active learning.

Active Learning. We adopted an iterative active learning loop and carried out **two** iterations, as validation gains plateaued and our annotation budget was fixed (pseudocode in Algorithm 1, Appendix B.1). Starting from the initial GPT-4o model M_0, fine-tuned on the 62-example seed set D_{seed}, each iteration proceeded as follows:

I) **Batch selection**: We randomly sampled descriptions, yielding batches B_{raw} of 94 instances in round 1 and 204 in round 2.

II) **Draft generation**: The current model M_i produced a scene graph draft for every selected description in the batch.

III) **Two-stage review**: The student annotator corrected each draft and the postdoc validated it, yielding refined graphs B_{refined}, which were then added to the training set ($D_{\text{train}} = D_{\text{train}} \cup B_{\text{refined}}$).

IV) **Model update**: GPT-4o was fine-tuned on the updated training set D_{train} to obtain the next model M_{i+1}.

After two rounds, the training set size $|D_{\text{train}}|$ increased from 62 to 360 examples while the annotation time per instance fell from ∼30 min to 10 min. Correspondingly, GPT-4o's SPICE on the validation set D_{val} increased from 70.8 to 73.3 to 76.1. We merged D_{train} and D_{val} and created two 300/100 train-test splits of the 400 human-annotated graphs.

2.3 Synthesis Annotation Pipeline

To further scale the data, we first used GPT-4o to filter out descriptions that showed hallucination errors when compared with their images, retaining 8,430 high-quality descriptions. We then used the teacher model, a GPT-4o fine-tuned on 300 human-annotated training examples, to generate

Dataset	# Inst.	Avg Len	Avg Trp	Avg Obj	Avg Rel	Total Trp
VG	2,966,195	5.34	1.53	1.69	1.22	4,533,271
FACTUAL	40,369	6.08	1.76	2.12	1.57	71,124
TSGBench	2,034	12.23	5.81	5.63	3.65	11,820
DiscoSG-DS						
Human	400	181.15	20.49	10.11	6.54	8,195
Synthetic	8,430	163.07	19.41	10.06	6.39	163,640

Table 2: Comparison of dataset statistics. Columns detail the number of instances (# Inst.), average description length in words (Avg Len), average counts per instance for triples (Avg Trp), unique objects (Avg Obj), and unique relations (Avg Rel), along with the total number of triples (Total Trp).

scene graph annotations for the remaining descriptions. The teacher model achieves a high SPICE score (F1 triple matching) of 73% on the test set.

2.4 Dataset Statistics and Analysis

The resulting DiscoSG-DS dataset comprises 300 human-annotated training and 100 test instances, along with 8,430 synthesised examples. Each instance includes an average of 9.3 sentences. Table 2 compares DiscoSG-DS with other datasets that contain **only single-sentence** captions, including Visual Genome (VG) (Krishna et al., 2017), FACTUAL (Li et al., 2023a), and TSGBench (Yang et al., 2025). On average, each DiscoSG-DS graph contains roughly 15× more triples than those in VG or FACTUAL and 3× than those in TSGBench, and demonstrates greater lexical diversity across longer spans of caption text (see Appendix C), highlighting its discourse-level complexity.

3 DiscoSG-Refiner: Iterative Graph Refinement for Scene Graph Parsing

Formally, given a multi-sentence image description $\boldsymbol{x}$, the goal of DiscoSG is to learn a parameterised model that predicts the most probable scene graph $\boldsymbol{y}' = \text{argmax}_{\boldsymbol{y}\in\mathcal{Y}} P(\boldsymbol{y} \mid \boldsymbol{x}; \boldsymbol{\theta})$, where $\mathcal{Y}$ is the set of all possible scene graphs. Existing approaches include fine-tuning small PLMs (Li et al., 2023a; Sharifzadeh et al., 2022), few-shot prompting with large PLMs (Yang et al., 2025), or fine-tuning large PLMs such as our teacher model. However, these methods either perform poorly on discourse-level inputs or incur high computational costs. The approach of merging sentence-level parser outputs (Dong et al., 2024) offers a better performance-efficiency trade-off but still fails to capture inter-sentence dependencies.

We propose DiscoSG-Refiner, a novel iterative refinement method using small open-source PLMs that mirrors our human annotation workflow. Inspired by machine translation post-editing (Vu and Haffari, 2018; Li et al., 2024) and text-to-graph generation (Han et al., 2024), it first produces a flawed initial parse and then applies targeted edits to yield the final scene graph, dramatically reducing the required generation overhead.

3.1 Framework Overview

DiscoSG-Refiner extends the Generator-Programmer-Interpreter framework (Vu and Haffari, 2018; Li et al., 2024) with task-specific adaptations across its three modules, placing particular focus on a novel Programmer optimised for refining long discourse-level scene graphs. The process begins with an initial graph $\boldsymbol{y}^0$ generated by the Generator. Then, over T steps, the Programmer proposes actions $\boldsymbol{a}^t$ which a deterministic Interpreter applies to produce $\boldsymbol{y}^{t+1}$, iterating until the final graph $\boldsymbol{y} = \boldsymbol{y}^T$ is obtained.

Formally, the probability of generating the final graph $\boldsymbol{y} = \boldsymbol{y}^T$ given the input $\boldsymbol{x}$ is defined as:

$$P(\boldsymbol{y}^T|\boldsymbol{x};\boldsymbol{\theta}) = P_{\text{Gen}}(\boldsymbol{y}^0|\boldsymbol{x}) \times \prod_{t=0}^{T-1} \left(P_{\text{Prog}}(\boldsymbol{a}^t|\boldsymbol{y}^t,\boldsymbol{x};\boldsymbol{\theta}) \cdot P_{\text{Intp}}(\boldsymbol{y}^{t+1}|\boldsymbol{y}^t,\boldsymbol{a}^t)\right) \quad (1)$$

3.1.1 Generator

The initial graph $\boldsymbol{y}^0$ probability is derived from applying a sentence-level parser P_{Sent} independently to each sentence $x_i \in \boldsymbol{x}$, such that $P_{Gen}(\boldsymbol{y}^0|\boldsymbol{x}) \propto \prod_{i=1}^{n} P_{\text{Sent}}(y_i|x_i)$. Operationally, we generate $\boldsymbol{y}^0$ using FACTUAL-T5 as P_{Sent} to parse each x_i into sentence-level graphs $\{y_1, \ldots, y_n\}$, which are then merged (treating nodes with identical names as coreferent) to form the initial graph. This provides broad coverage of explicit mentions but often introduces errors of commission (redundancies) and omission (missing cross-sentence or implicit relations), requiring subsequent refinement.

3.1.2 Programmer

At each step t, Programmer ($P_{\text{Prog}}(\boldsymbol{a}^t|\boldsymbol{y}^t,\boldsymbol{x};\boldsymbol{\theta})$), our core learnable component with parameters $\boldsymbol{\theta}$, generates a set of edit actions $\boldsymbol{a}^t = (\mathbf{D}^t, \mathbf{I}^t)$, which includes deletion $\mathbf{D}^t$ and insertion $\mathbf{I}^t$ operations based on the current graph $\boldsymbol{y}^t$ and the input $\boldsymbol{x}$.

Directly applying previous post-editing techniques is suboptimal. Methods generating combined deletion/insertion sequences (Li et al., 2024) require overly long generation for small PLM decoders, while insert-only methods (Han et al., 2024), although having shorter generation length, cannot correct the errors of commission present in

our initial $\boldsymbol{y}^0$. We observe that approximately 1/3 of the initial graph triples require deletion.

Therefore, we propose a **novel** Programmer architecture using an encoder-decoder PLM (e.g., Flan-T5) that *disentangles deletion and insertion prediction to reduce decoder generation length*:

Deletion Prediction (Encoder-based): The encoder takes $\boldsymbol{x}$ and $\boldsymbol{y}^t$ as input. The objective is to identify which triples to delete, with each triple represented by the average of contextual representations of tokens between its opening and closing brackets. Specifically, the encoder predicts `KEEP[`t_i`]` (0) or `DELETE[`t_i`]` (1) action flags through a binary classification layer for each triple t_i in the flattened representation of $\boldsymbol{y}^t$.

Insertion Generation (Decoder-based): The decoder, conditioned on the encoder's representation of $\boldsymbol{x}$ and $\boldsymbol{y}^t$, generates token sequences forming *unseen graph triples*, each generated token t_j implicitly representing an `INSERT[`t_j`]` action. This approach significantly reduces generation length compared to regenerating the entire graph.

3.1.3 Interpreter

The Interpreter ($P_{\text{Intp}}(\boldsymbol{y}^{t+1}|\boldsymbol{y}^t, \boldsymbol{a}^t)$) applies the edit actions $\boldsymbol{a}^t = (\mathbf{D}^t, \mathbf{I}^t)$ to transform $\boldsymbol{y}^t$ into $\boldsymbol{y}^{t+1}$. Because this process is deterministic, the probability $P_{\text{Intp}}(\boldsymbol{y}^{t+1}|\boldsymbol{y}^t, \boldsymbol{a}^t)$ is simply 1 for the unique resulting state $\boldsymbol{y}^{t+1}$ (and 0 otherwise). The Interpreter operates in two stages:

Deletion: With deletion flags $\mathbf{D}^t$, the Interpreter removes the identified triples $\mathcal{D}^t_{\text{triples}}$ from $\boldsymbol{y}^t$.

Insertion: The token sequence $\mathbf{I}^t$ generated by the decoder is parsed into candidate triples. The Interpreter then validates these for structural correctness (e.g., subject-relation-object format) and filters out any malformed or incomplete ones. This yields the set of valid triples $\mathcal{I}^t_{\text{triples}}$, which are subsequently inserted into $\boldsymbol{y}^t$ as an unordered collection.

The resulting graph state for the next iteration follows a delete-first-then-insert heuristic: $\boldsymbol{y}^{t+1} = (\boldsymbol{y}^t \setminus \mathcal{D}^t_{\text{triples}}) \cup \mathcal{I}^t_{\text{triples}}$.

3.2 Training the Programmer

Edit-Annotated Data (DiscoSG-ED) Generation. Training requires editing annotations derived from DiscoSG-DS. For each instance $(\boldsymbol{x}, \boldsymbol{y}_{gold})$ from both the manual and synthetic sets, we generate the initial graph $\boldsymbol{y}^0$ given $\boldsymbol{x}$ using Generator. We also create alternative versions of $\boldsymbol{y}^{(0)}$ by synthetically corrupting $\boldsymbol{y}_{\text{gold}}$ via the random deletion or insertion of graph triples. The ground-truth edit actions $(\mathbf{D}_{gt}, \mathbf{I}_{gt})$ are derived by comparing both types of $\boldsymbol{y}^0$ against $\boldsymbol{y}_{gold}$.

The ground-truth deletion flags $\mathbf{D}_{gt}$ are derived by finding the set of triples $\mathcal{D}_{gt}$ present in $\boldsymbol{y}^0$ but not in $\boldsymbol{y}_{gold}$. For insertion, after identifying triples $\mathcal{I}_{gt}$ present in $\boldsymbol{y}_{gold}$ but not in $\boldsymbol{y}^0$, we convert triples into token sequences to form the ground-truth insertions $\mathbf{I}_{gt}$ to train the decoder. These derived tuples $(\boldsymbol{x}, \boldsymbol{y}^0, \mathbf{D}_{gt}, \mathbf{I}_{gt})$ form our training dataset, DiscoSG-ED. If we generate N corrupted versions per gold graph, the size of DiscoSG-ED becomes N times that of DiscoSG-DS.

Loss Functions and Optimisation. The Programmer's encoder and decoder are jointly trained via multitask learning, optimising separate supervised objectives. The encoder is trained using a binary cross-entropy loss ($\mathcal{L}_{\text{delete}}$). The decoder is trained using a standard sequence-to-sequence cross-entropy loss ($\mathcal{L}_{\text{insert}}$) to generate the target insertion sequence $\mathbf{I}_{gt}$. The final loss is a weighted sum ($\mathcal{L} = \lambda_{\text{del}}\mathcal{L}_{\text{delete}} + \lambda_{\text{ins}}\mathcal{L}_{\text{insert}}$).

4 Experiments

We evaluate DiscoSG-Refiner through three complementary analyses: (i) benchmark performance on the DiscoSG-DS test set, (ii) impact on downstream parsing tasks, and (iii) ablation studies to isolate the contribution of each component.

4.1 Discourse-Level Text Scene Graph Parsing Evaluation

Dataset. We evaluate on the DiscoSG-DS **Random** and **Length** test sets (100 each). *Random* is a uniform split, while *Length* sorts by the number of graph triples, so training covers fewer-triple graphs and testing targets more complex graphs, assessing generalisation to long, complex graphs.

Metrics. Parser performance is measured by comparing the predicted scene graph against the gold graph using these metrics: 1) **SPICE** (Anderson et al., 2016), an F1 score for exact-match triple overlap; 2) **Bi-SoftSPICE (BSSPICE)**, a symmetrized version of SoftSPICE (Li et al., 2023a) using semantic similarities between triple embeddings measuring graph similarities; 3) **computational efficiency metrics** (i.e., inference time[2] and model parameter count); and 4) **discourse-specific**

[2]For methods using models ≥70B parameters and GPT-4o, inference time is measured via API call time per query.

error rate (%), measuring four types of discourse-level parsing errors based on GPT-4o and human assessments. For each instance, if an error type is present in the graph, it is counted as one error regardless of how many times it occurs. Detailed error definitions are provided in Section A.3.

Methods Compared. We compare DiscoSG-Refiner against 21 methods across 5 categories.

(I) Sentence Parsing & Merging methods parse each sentence in a multi-sentence caption independently, then merge the resulting scene graphs. The parsers include Stanford-Parser (Schuster et al., 2015), as used in the original SPICE implementation (Anderson et al., 2016), as well as VG-T5-base (Sent+Merge) (Sharifzadeh et al., 2022) and FACTUAL-T5-base (Sent+Merge) (Li et al., 2023a), which use Flan-T5-base models fine-tuned on sentence-level data from VG and FACTUAL.

(II) End-to-End (Sentence) methods, including VG-T5-base (Direct) and FACTUAL-T5 (Direct), apply these sentence-trained T5 models directly to full discourse inputs without intermediate merging.

(III) End-to-End (Discourse) methods include DiscoSG-T5 (base/large/xl) and DiscoSG-Qwen2.5 (0.5B/1.5B/7B parameters), which are Flan-T5 and Qwen2.5 (Bai et al., 2023) variants fine-tuned on the full DiscoSG-DS training set (8,730 examples). We also report DiscoSG-GPT-4o (Teacher) and DiscoSG-T5-large (300), i.e., GPT-4o and Flan-T5-large fine-tuned on 300 expert-curated examples only.

(IV) Few-Shot Prompting baselines use large instruction-tuned PLMs, including Qwen2.5-72B-Instruct, Llama-3.3-70B-Instruct (Grattafiori et al., 2024), and GPT-4o (text-only and multimodal). We use 3-shot prompts, each a caption-graph pair from DiscoSG-DS. We also report a 3-shot GPT-4o-based GraphRAG (Edge et al., 2024) variant, where each shot includes the caption, extracted objects and relations, and the reference graph.

(V) Iterative Refinement methods, including PiVe (Han et al., 2024), Self-Refine (Madaan et al.), and Prog-Refine (Li et al., 2024), all employ 3-shot GPT-4o as both Generator and Interpreter, differing only in their Programmer implementations.

All baseline details are in Section A.1.

4.1.1 Results and Analysis

DiscoSG-DS often enhances discourse parsing, with gains depending on model capacity, architecture, and training strategy. As shown in Table 3, few-shot Llama-3.3-70B-Instruct reaches 52.0/53.5 SPICE on Random/Length, surpassing FACTUAL-T5-base (Sent+Merge) (49.4/52.7), the strongest baseline *without using DiscoSG-DS*. Larger gains come from fine-tuning: the encoder-decoder DiscoSG-T5-xl (3B) attains 76.8/66.0 SPICE (+55% over 49.4 on Random; +25% over 52.7 on Length), the decoder-only DiscoSG-GPT-4o achieves 73.1/74.5 (+48%/+41% over the respective baselines), and our DiscoSG-Refiner-base (two fine-tuned Flan-T5-base models) yields 64.3/67.3 SPICE, about **+30%** over the sentence-merging baselines on both splits.

However, discourse-level supervision alone is insufficient: robust performance across graph complexities requires either higher-capacity PLMs of the right architecture *with* fine-tuning, or specialised procedures such as our iterative refinement. For example, within encoder-decoder models, scaling with fine-tuning yields the largest improvements, whereas naive end-to-end fine-tuning of small PLMs generalises poorly and is sensitive to graph complexity: DiscoSG-T5-base (0.25B) and -large (0.78B) score 52.7 and 69.4 on Random but drop to 38.4 and 53.0 on Length, respectively. Decoder-only fine-tuned PLMs also underperform on Length unless scaled to GPT-4o-level sizes. DiscoSG-Qwen2.5 at 0.5B, 1.5B, and 7B yields SPICE scores of 48.5, 51.6, and 47.3, each below the 52.7 baseline on Length split.

DiscoSG-Refiner achieves the best balance of performance, generalisation, and cost. DiscoSG-Refiner models of all sizes outperform all other baselines on *Length*, including those requiring vastly more computational resources ($\geq$70B vs. our **max 3.25B** parameters), except for DiscoSG-GPT-4o. Nevertheless, DiscoSG-Refiner-large scores around 7 SPICE and 1.2 BSSPICE points below DiscoSG-GPT-4o on both splits while being **50**$\times$ faster, and our base version achieves an **86**$\times$ speedup. Beyond speed, our approach offers significant cost benefits: running DiscoSG-GPT-4o on image captioning benchmarks DetailCaps (Dong et al., 2024) and CapArena (Cheng et al., 2025) would cost an estimated **$25-150** in API fees, whereas all versions of our approach can execute locally on a single RTX 4090, eliminating API costs and network latency. DiscoSG-GPT-4o cannot be released due to licensing restrictions, while our models are open-source. Compared to similar-cost methods like DiscoSG-T5 variants, our approach demonstrates superior generalisation: DiscoSG-T5 models perform well on Random but suffer degra-

Category	Method	Random SPICE↑	Random BSSPICE↑	Length SPICE↑	Length BSSPICE↑	Time (s)↓	Param.↓
Sentence Parsing & Merging	Stanford-Parser	17.0	81.5	19.5	83.1	~0.21s	-
	VG-T5-base (Sent+Merge)	45.3	89.4	45.9	90.1	~0.19s	0.25B
	FACTUAL-T5-base (Sent+Merge)	49.4	90.9	52.7	92.0	~0.19s	0.25B
End-to-End (Sentence)	VG-T5-base (Direct)	15.3	82.5	13.8	82.4	~0.07s	0.25B
	FACTUAL-T5 (Direct)	37.6	88.0	32.2	87.5	~0.07s	0.25B
End-to-End (Discourse) ***DiscoSG-DS (8730 examples)***	DiscoSG-T5-base	52.7	91.7	38.4	88.5	~0.07s	0.25B
	DiscoSG-T5-large (8730)	69.4	95.1	53.0	91.8	~0.32s	0.78B
	DiscoSG-T5-xl	76.8	96.8	66.0	94.9	~0.39s	3B
	DiscoSG-Qwen2.5-0.5B	54.2	90.1	48.5	90.0	~0.39s	0.5B
	DiscoSG-Qwen2.5-1.5B	65.2	94.3	51.6	89.5	~0.77s	1.5B
	DiscoSG-Qwen2.5-7B	20.8	75.0	47.3	90.6	~3.27s	7B
End-to-End (Discourse) ***DiscoSG-DS (300 examples)***	DiscoSG-T5-large (300)	45.4	90.1	34.2	87.8	~0.71s	0.78B
	DiscoSG-GPT-4o (Teacher)	73.1	96.0	74.5	96.4	~36.9s	-
Few-Shot Prompting ***(3-shot)***	Qwen2.5-72B-Instruct	48.0	90.2	50.1	91.2	~10.4s	72B
	Llama-3.3-70B-Instruct	52.0	91.6	53.5	92.4	~14.1s	70B
	GPT-4o (text-only)	53.2	91.7	52.5	92.0	~19.8s	-
	GPT-4o (multimodal)	55.6	92.3	54.4	92.4	~33.2s	-
	GraphRAG-GPT-4o (text-only)	47.7	90.9	41.4	90.0	~71.7s	-
Iterative Refinement ***Prior Methods***	PiVe	54.0	92.0	52.1	92.0	~60.8s	-
	Self-Refine	37.2	86.2	34.6	83.5	~99.0s	-
	Prog-Refine	53.4	90.9	52.5	92.1	~60.8s	-
Iterative Refinement ***Our Method***	DiscoSG-Refiner-base	64.3	94.3	67.3	95.1	~0.43s	0.5B
	DiscoSG-Refiner-large	66.2	94.7	67.3	95.2	~0.71s	1.03B
	DiscoSG-Refiner-xl	**66.7**	**94.9**	**68.6**	**95.4**	~2.17s	3.25B

Table 3: Comparison of methods on DiscoSG-DS Random and Length test sets. **Bold** values denote the best results among our method variants, while underlined values show overall top performance.

	Cross.↓	Long.↓	Impl.↓	Graph.↓	Avg.↓
FACTUAL-T5-base (Sent+Merge)	61.0	97.0	77.0	93.0	82.00
DiscoSG-T5-base	67.0	100.0	96.0	100.0	90.75
GPT-4o (text-only)	36.0	81.0	83.0	65.0	66.25
DiscoSG-GPT-4o (Teacher)	28.0	86.0	76.0	81.0	67.75
DiscoSG-Refiner-large	43.0	85.0	74.0	79.0	70.25
Average	47.0	89.8	81.2	83.6	75.4

Table 4: Discourse error rates (%) identified by GPT-4o-based annotation across four error categories: Cross-Sentence Coreference Resolution (Cross.), Long-Range Relational Dependency (Long.), Implicit Information Inference (Impl.), and Graph Coherence (Graph.). See human evaluation results in Appendix A.3.

dation on Length (11-16 SPICE point drops). In contrast, our models maintain high performance (64 to 68 SPICE) across both test splits.

Cross-sentence coreference resolution is the least difficult among the discourse parsing challenges. Table 4 highlights clear differences across error types in parser outputs. Parsers achieve the lowest error rates regarding linking entities across sentences, ranging from 28% to 61%. In contrast, the other three categories prove far more difficult, with rates clustering around 80% to 90%. The small DiscoSG-T5-base model shows the highest errors in every category and even exceeds the sentence-level baseline, confirming that small PLMs require specialised designs for discourse tasks. Overall, existing methods identify coreferent entities reasonably well, yet still struggle to maintain relational consistency, recover implicit information, and preserve graph coherence.

4.2 Evaluation on Downstream VLM Tasks

We evaluate the impact of different automated metrics on image captioning assessment and hallucination detection across three benchmark datasets.

Benchmarks. For **Image Captioning**, we use CapArena (Cheng et al., 2025), which includes 6,000 human-written reference captions, each compared against outputs from 14 VLMs based on human preferences, and used to rank VLM performance. DetailCaps (Dong et al., 2024) provides 4,870 images with captions rated 1-5 by GPT-4o. For **Hallucination Detection**, we introduce D-FOIL, a new benchmark designed to evaluate how well different metrics detect hallucinations in discourse-level outputs from VLMs. Inspired by FOIL (Shekhar et al., 2017), D-FOIL contains 200 multi-sentence captions from SharedGPT4V with subtle hallucinated entities or relations. Each hallucinated caption is paired with a minimally corrected version annotated by humans, along with a reference caption generated by GPT-4.1. See Section B.3 for collection details.

Metrics. We evaluate using **graph-based metrics** including SPICE, BSSPICE, and CAPTURE (Dong et al., 2024), which measure sim-

Metric	DetailCaps τ ↑	DetailCaps ρ ↑	CapArena τ ↑	CapArena ρ ↑	CapArena Acc.↑	D-FOIL Acc.↑	Avg. Rank↓
Token Length	8.8	12.2	58.2	71.0	68.3	2.00	12.0
N-gram							
BLEU-4	25.6	36.4	31.9	42.4	47.4	53.8	14.3
METEOR	27.7	39.4	56.0	77.1	61.2	61.3	7.5
ROUGE-L	23.6	33.4	1.1	-6.4	48.0	50.2	17.7
CIDEr	24.7	35.1	-27.9	-20.9	38.4	52.0	19.5
Embedding-based							
RefCLIPScore	30.4	44.6	-45.1	-57.4	32.5	37.5	19.0
Polos	34.8	53.3	36.3	42.0	47.9	75.5	10.8
VLM-based							
FLEUR	-3.5	-2.0	29.7	39.3	45.8	18.0	19.3
GPT-4o Eval	-	-	94.3	84.6	62.7	91.5	-
Graph-based							
SPICE w/							
FACTUAL (S+M)†	40.7	55.1	36.3	46.8	52.8	**86.5**	8.2
DiscoSG-T5-base	25.4	35.2	23.1	36.7	37.7	42.0	17.8
DiscoSG-T5-large	31.8	43.5	3.3	9.9	42.4	53.0	15.8
DiscoSG-Refiner-base	41.9	56.6	38.5	47.3	53.0	**86.5**	6.7
DiscoSG-Refiner-large	**42.2**	**56.9**	**40.7**	**50.8**	**53.5**	**86.5**	**5.8**
BSSPICE w/							
FACTUAL (S+M)†	38.8	56.6	49.5	62.2	54.5	88.0	6.2
DiscoSG-T5-base	25.2	36.3	34.1	45.5	47.1	49.5	15.3
DiscoSG-T5-large	30.6	43.9	7.7	13.0	47.5	57.5	14.0
DiscoSG-Refiner-base	40.0	58.0	51.6	66.2	55.2	87.5	5.0
DiscoSG-Refiner-large	**40.3**	**58.5**	**53.8**	**70.5**	**55.4**	**89.0**	**3.7**
CAPTURE w/							
FACTUAL (S+M)†	39.6	54.5	53.8	61.3	57.6	69.5	7.2
DiscoSG-T5-base	18.0	26.0	-3.3	5.1	45.1	43.5	19.8
DiscoSG-T5-large	22.7	32.5	-18.7	-17.8	46.1	55.0	18.7
DiscoSG-Refiner-base	40.7	55.9	51.6	60.9	**58.4**	71.0	6.2
DiscoSG-Refiner-large	**40.8**	**56.0**	**58.2**	**74.1**	58.2	**74.0**	**4.0**

Table 5: Downstream task evaluation results. †FACTUAL-T5-base (Sent+Merge).

ilarity between the scene graphs of candidate and reference captions. Specifically, we compare these metrics when using scene graphs parsed by DiscoSG-Refiner versus those from the FACTUAL-T5-base (Sent+Merge) baseline. **Other metrics include:** RefCLIPScore (Hessel et al., 2021) and POLOS (Wada et al., 2024) (embedding-based), FLEUR (Lee et al., 2024) and zero-shot GPT-4o (VLM-based), BLEU-4 (Papineni et al., 2002), METEOR (Banerjee and Lavie, 2005), ROUGE-L (Lin, 2004), and CIDEr (Vedantam et al., 2015) (n-gram), and caption length (longer captions preferred).

Evaluation Setup. For image captioning, we compute Kendall's τ and Spearman's ρ correlations between metric scores and gold-standard judgments (GPT-4o ratings for DetailCaps, human preferences for CapArena). For CapArena, we also report agreement accuracy (Acc.), defined as the percentage of metric predictions that match human preferences. For hallucination detection on D-FOIL, we report classification accuracy (Acc.) in distinguishing hallucinated captions from their corrected versions. Finally, we compute the average performance rank of each metric (excluding GPT-4o Eval) by ordering methods by correlation or accuracy across all evaluation scenarios.

4.2.1 Results and Analysis

Scene graphs from DiscoSG-Refiner enhance graph-based metrics across tasks. Table 5 shows that replacing FACTUAL-T5-base (Sent+Merge) with DiscoSG-Refiner-large improves SPICE, raising Kendall's τ from 40.7 to 42.2 on DetailCaps and from 36.3 to 40.7 on CapArena. BSSPICE also improves, with τ rising from 38.8 to 40.3 on DetailCaps, from 49.5 to 53.8 on CapArena, and classification accuracy increasing from 88.0% to 89.0% on D-FOIL. CAPTURE increases from 53.8 to 58.2 in τ on CapArena and from 69.5% to 74.0% in accuracy on D-FOIL. Even the smaller DiscoSG-Refiner-base surpasses the baseline in nearly all settings, and, on average, replacing the sentence-level parser with DiscoSG-Refiner improves the rank of each graph-based metric by about one to two positions. In contrast, replacing FACTUAL-T5-base (Sent+Merge) with DiscoSG-T5 parsers reduces performance for **all** graph-based metrics across all tasks. These results indicate that our refinement architecture produces higher-quality scene graphs, which in turn enable more accurate evaluation of image captioning and hallucination detection.

Graph-based metrics with DiscoSG-Refiner-large achieve top performance among all automated metrics. BSSPICE attains the best average rank of 3.7, closely followed by CAPTURE at 4.0. These results outperform commonly used metrics such as METEOR, with an average rank of 7.5, and RefCLIPScore, which ranks 19.0. On D-FOIL, BSSPICE achieves 89.0 percent accuracy, second only to GPT-4o Eval at 91.5 percent, while METEOR and RefCLIPScore reach only 61.3 and 37.5 percent, respectively. This indicates that graph-based metrics are more effective at detecting subtle semantic issues, such as incorrect entities or relations, which are often missed by n-gram or embedding-based metrics.

4.3 Ablation Study

Evaluation Setup. Table 6 probes three design axes of DiscoSG-Refiner-large. The *Default* row derives DiscoSG-ED from the full DiscoSG-DS (300 expert + 8,430 synthetic instances) with 15 × augmentation, executes two refinement iterations, and employs a disentangled *Programmer* (encoder-deletion, decoder-insertion). *Each variant changes exactly one of these factors while holding the others fixed.* **(I) Training-data source**: *Expert-only* DiscoSG-ED *15*× removes the synthetic portion

System / Condition	DiscoSG-DS (Random) SPICE↑	BSSPICE↑	DetailCaps τ ↑	ρ ↑
DiscoSG-Refiner-large (Default)	66.2	94.7	42.2	56.9
Ablation: Training–data source / augmentation				
Expert-only (300) DiscoSG-ED 15×	52.1	91.8	42.1	56.9
Expert+Synth. (8730) DiscoSG-ED 1×	62.9	94.0	41.9	56.5
Expert+Synth. (8730) DiscoSG-ED 5×	64.4	94.3	41.8	56.5
Ablation: Refinement depth				
1 iteration	63.5	94.2	42.1	56.8
3 iterations	66.7	94.8	42.3	57.1
Ablation: Programmer edit modality				
Deletion-only	48.9	90.4	41.9	56.7
Insertion-only	53.4	92.1	42.1	56.7
Ablation: Action-generation architecture				
Monolithic (no disentangle)	51.9	91.6	41.7	55.4

Table 6: Ablation studies on DiscoSG-Refiner-large.

of DiscoSG-DS. **(II) Augmentation scale**: using the full DiscoSG-DS data, we reduce the number of edit instances by setting the augmentation multiplier to 1× or 5×. **(III) Refinement depth**: we run a single or three iterations. **(IV) Programmer edit modality**: we activate only deletion or only insertion edits. **(V) Action-generation architecture**: a *Monolithic Encoder-Decoder* variant replaces the disentangled design, using one decoder for both edit types. Results are reported with SPICE and BSSPICE on the DiscoSG-DS test, and SPICE's Kendall's τ / Spearman's ρ on DetailCaps.

4.3.1 Results and Analysis

Synthetic data, augmentation scale, refinement depth, and disentangled edit actions are the main drivers of performance. **I)** Using *Expert-only DiscoSG-ED 15×* lowers SPICE by 14 points, confirming that synthetic examples in DiscoSG-DS add crucial graph coverage beyond what expert data provides. **II)** Expanding augmentation from 1× to 15× yields steady yet diminishing SPICE gains (62.9 to 66.2) and marginal BSSPICE improvements, indicating that moderate edit operation expansion is already beneficial. **III)** Increasing refinement iterations from 1 to 3 boosts all metrics, with 3 passes achieving the overall best scores. We default to 2 mainly for efficiency. **IV)** Activating only deletion or insertion edits reduces SPICE by 12 to 17 points, showing the importance of their combination for good graph correction. **V)** Replacing the disentangled Programmer with a monolithic encoder-decoder degrades performance, as generating long, combined edit sequences proves too challenging for Flan-T5-large decoder.

5 Related Work

Scene Graph Parsing Methods. Scene graph parsing has been explored in images (Zellers et al., 2018; Tang et al., 2020; Xu et al., 2017; Zhang et al., 2019; Cong et al., 2022; Li et al., 2022; Im et al., 2024; Wu et al., 2025b), video (Rodin et al., 2024), and multimodal inputs (Wu et al., 2025a). Our work focuses on text-based parsing, where two primary strategies exist. The most common approach uses direct generation: encoder-decoder models are fine-tuned on datasets like VG (Wang et al., 2018; Choi et al., 2022) and FACTUAL (Li et al., 2023a), or PLMs are prompted with few-shot examples from benchmarks like TSGBench (Yang et al., 2025). The second strategy converts text into an intermediate representation, such as dependency parses or AMR, which is then transformed into a scene graph (Schuster et al., 2015; Anderson et al., 2016; Choi et al., 2022). Existing datasets like FACTUAL enable strong sentence-level parsing but focus on isolated sentences, causing trained parsers to overlook discourse context.

Downstream Applications. Scene graphs have been successfully applied in vision-language tasks. They improve image retrieval (Johnson et al., 2015; Andrews et al., 2019) and guide structured image caption generation (Zhong et al., 2020; Zeng et al., 2024). Scene graphs also serve as a foundational representation for evaluating caption factual accuracy via metrics like SPICE (Anderson et al., 2016), SoftSPICE (Li et al., 2023a), and CAPTURE (Dong et al., 2024). They are also used for identifying and mitigating hallucinations in VLM outputs (Yu et al., 2024).

6 Conclusion

We introduce DiscoSG, a new task for discourse-level text scene graph parsing, along with DiscoSG-DS, a dataset of 8,830 VLM-generated captions annotated with semantically coherent scene graphs by human experts and fine-tuned GPT-4o, where the graphs capture key discourse phenomena. To achieve high performance at low cost, we propose DiscoSG-Refiner, an efficient iterative refinement framework that uses small PLMs for initial graph generation and then refines graphs through triple insertion and deletion. Our smallest model achieves a 30% SPICE improvement over the best sentence-level baseline on two test splits while significantly reducing inference costs compared to the GPT-4o parser. We demonstrate the framework's effectiveness on downstream vision-language tasks and introduce D-FOIL, a benchmark for evaluating hallucination detection in discourse-level VLM outputs.

Limitations

Despite including images in our dataset, our current approach operates in a text-only setting for discourse-level scene graph parsing. All parsing methods except GPT-4o (multimodal) rely solely on textual descriptions without incorporating visual cues from the original images. While this presents an opportunity for future work, our evaluation of SOTA multimodal parsers like GPT-4o (multimodal), as well as findings from Universal Scene Graph Generation (Wu et al., 2025a), suggest that current models incorporating additional visual information do not yield significant improvements over the best text-only approaches. This indicates that effectively leveraging multimodal information to boost text scene graph parsing still remains challenging.

Acknowledgments

This work is a collaboration across Wuhan University, Monash University, and RMIT University. We thank Wuhan University for access to GPU resources. OpenAI API costs were primarily paid from Zhuang Li's personal account.

Author Contributions

- **Zhuang Li (RMIT University; senior author).** Led the core idea and methodology; authored the main draft and edited most subsequent drafts; oversaw project execution and advised the student on this project (informal supervision); contributed to and oversaw data annotation (e.g., graph validation and student annotator training); conducted a small set of experiments; covered most API fees.
- **Shaoqing Lin (Wuhan University; student author).** Led experimental investigation and performed most data annotation; contributed additional ideas and methodological details; contributed to manuscript drafting and review.
- **Chong Teng (Wuhan University; corresponding author).** Managed project administration and correspondence; student's supervisor; coordinated access to computational resources and GPU.
- **Donghong Ji (Wuhan University).** Provided computational resources and GPU access; co-supervises the student.
- **Lizhen Qu (Monash University).** Provided discussions and high-level suggestions.
- **Fei Li (Wuhan University).** Co-supervises the student.

References

Peter Anderson, Basura Fernando, Mark Johnson, and Stephen Gould. 2016. Spice: Semantic propositional image caption evaluation. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part V 14*, pages 382–398. Springer.

Martin Andrews, Yew Ken Chia, and Sam Witteveen. 2019. Scene graph parsing by attention graph. *arXiv preprint arXiv:1909.06273*.

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, and 1 others. 2023. Qwen technical report. *arXiv preprint arXiv:2309.16609*.

Satanjeev Banerjee and Alon Lavie. 2005. Meteor: An automatic metric for mt evaluation with improved correlation with human judgments. In *Proceedings of the acl workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization*, pages 65–72.

Lin Chen, Jinsong Li, Xiaoyi Dong, Pan Zhang, Conghui He, Jiaqi Wang, Feng Zhao, and Dahua Lin. 2024. Sharegpt4v: Improving large multi-modal models with better captions. In *European Conference on Computer Vision*, pages 370–387. Springer.

Kanzhi Cheng, Wenpo Song, Jiaxin Fan, Zheng Ma, Qiushi Sun, Fangzhi Xu, Chenyang Yan, Nuo Chen, Jianbing Zhang, and Jiajun Chen. 2025. CapArena: Benchmarking and analyzing detailed image captioning in the LLM era. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 14077–14094, Vienna, Austria. Association for Computational Linguistics.

Woo Suk Choi, Yu-Jung Heo, Dharani Punithan, and Byoung-Tak Zhang. 2022. Scene graph parsing via Abstract Meaning Representation in pre-trained language models. In *Proceedings of the 2nd Workshop on Deep Learning on Graphs for Natural Language Processing (DLG4NLP 2022)*, pages 30–35, Seattle, Washington. Association for Computational Linguistics.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, and 1 others. 2024. Scaling instruction-finetuned language models. *Journal of Machine Learning Research*, 25(70):1–53.

Yuren Cong, Michael Ying Yang, and Bodo Rosenhahn. 2022. Reltr: Relation transformer for scene graph generation. *arXiv preprint arXiv:2201.11460*.

Michael A Covington and Joe D McFall. 2010. Cutting the gordian knot: The moving-average type–token ratio (mattr). *Journal of quantitative linguistics*, 17(2):94–100.

Hongyuan Dong, Jiawen Li, Bohong Wu, Jiacong Wang, Yuan Zhang, and Haoyuan Guo. 2024. Benchmarking and improving detail image caption. *arXiv preprint arXiv:2405.19092*.

Darren Edge, Ha Trinh, Newman Cheng, Joshua Bradley, Alex Chao, Apurva Mody, Steven Truitt, Dasha Metropolitansky, Robert Osazuwa Ness, and Jonathan Larson. 2024. From local to global: A graph rag approach to query-focused summarization. *arXiv preprint arXiv:2404.16130*.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, and 1 others. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Jiuzhou Han, Nigel Collier, Wray Buntine, and Ehsan Shareghi. 2024. Pive: Prompting with iterative verification improving graph-based generative capability of llms. In *Findings of the Association for Computational Linguistics ACL 2024*, pages 6702–6718.

Jack Hessel, Ari Holtzman, Maxwell Forbes, Ronan Le Bras, and Yejin Choi. 2021. Clipscore: A reference-free evaluation metric for image captioning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 7514–7528.

Jinbae Im, JeongYeon Nam, Nokyung Park, Hyungmin Lee, and Seunghyun Park. 2024. Egtr: Extracting graph from transformer for scene graph generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 24229–24238.

Justin Johnson, Ranjay Krishna, Michael Stark, Li-Jia Li, David Shamma, Michael Bernstein, and Li Fei-Fei. 2015. Image retrieval using scene graphs. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3668–3678.

Ranjay Krishna, Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen, Yannis Kalantidis, Li-Jia Li, David A Shamma, and 1 others. 2017. Visual genome: Connecting language and vision using crowdsourced dense image annotations. *International journal of computer vision*, 123(1):32–73.

Yebin Lee, Imseong Park, and Myungjoo Kang. 2024. Fleur: An explainable reference-free evaluation metric for image captioning using a large multimodal model. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3732–3746.

Rongjie Li, Songyang Zhang, and Xuming He. 2022. Sgtr: End-to-end scene graph generation with transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 19486–19496.

Zhuang Li, Yuyang Chai, Terry Yue Zhuo, Lizhen Qu, Gholamreza Haffari, Fei Li, Donghong Ji, and Quan Hung Tran. 2023a. Factual: A benchmark for faithful and consistent textual scene graph parsing. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 6377–6390.

Zhuang Li, Levon Haroutunian, Raj Tumuluri, Philip R Cohen, and Reza Haf. 2024. Improving cross-domain low-resource text generation through llm post-editing: A programmer-interpreter approach. In *Findings of the Association for Computational Linguistics: EACL 2024*, pages 347–354.

Zhuang Li, Lizhen Qu, Philip R Cohen, Raj Tumuluri, and Gholamreza Haffari. 2023b. The best of both worlds: Combining human and machine translations for multilingual semantic parsing with active learning. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9511–9528.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, and 1 others. Self-refine: Iterative refinement with self-feedback, 2023. *URL https://arxiv.org/abs/2303.17651*.

Philip M McCarthy. 2005. *An assessment of the range and usefulness of lexical diversity measures and the potential of the measure of textual, lexical diversity (MTLD)*. Ph.D. thesis, The University of Memphis.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992.

Ivan Rodin, Antonino Furnari, Kyle Min, Subarna Tripathi, and Giovanni Maria Farinella. 2024. Action scene graphs for long-form understanding of egocentric videos. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18622–18632.

Sebastian Schuster, Ranjay Krishna, Angel Chang, Li Fei-Fei, and Christopher D Manning. 2015. Generating semantically precise scene graphs from textual descriptions for improved image retrieval. In *Proceedings of the fourth workshop on vision and language*, pages 70–80.

Sahand Sharifzadeh, Sina Moayed Baharlou, Martin Schmitt, Hinrich Schütze, and Volker Tresp. 2022. Improving scene graph classification by exploiting knowledge from texts. In *Proceedings of the AAAI conference on artificial intelligence*, volume 36, pages 2189–2197.

Ravi Shekhar, Sandro Pezzelle, Yauhen Klimovich, Aurelie Herbelot, Moin Nabi, Enver Sangineto, Raffaella Bernardi, and 1 others. 2017. Foil it! find one mismatch between image and language caption. In *ACL 2017 The 55th Annual Meeting of the Association for Computational Linguistics: Proceedings of the Conference, Vol. 1 (Long Papers)*, pages 255–265. Association for Computational Linguistics (ACL).

Kaihua Tang, Yulei Niu, Jianqiang Huang, Jiaxin Shi, and Hanwang Zhang. 2020. Unbiased scene graph generation from biased training. In *Conference on Computer Vision and Pattern Recognition*.

Ramakrishna Vedantam, C Lawrence Zitnick, and Devi Parikh. 2015. Cider: Consensus-based image description evaluation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4566–4575.

Thuy-Trang Vu and Reza Haffari. 2018. Automatic post-editing of machine translation: A neural programmer-interpreter approach. In *Empirical Methods in Natural Language Processing 2018*, pages 3048–3053. Association for Computational Linguistics (ACL).

Yuiga Wada, Kanta Kaneda, Daichi Saito, and Komei Sugiura. 2024. Polos: Multimodal metric learning from human feedback for image captioning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13559–13568.

Yu-Siang Wang, Chenxi Liu, Xiaohui Zeng, and Alan Yuille. 2018. Scene graph parsing as dependency parsing. In *Proceedings of NAACL-HLT*, pages 397–407.

Shengqiong Wu, Hao Fei, and Tat-Seng Chua. 2025a. Universal scene graph generation. In *CVPR*.

Shengqiong Wu, Hao Fei, Jingkang Yang, Xiangtai Li, Juncheng Li, Hanwang Zhang, and Tat-Seng Chua. 2025b. Learning 4d panoptic scene graph generation from rich 2d visual scene. In *CVPR*.

Danfei Xu, Yuke Zhu, Christopher B Choy, and Li Fei-Fei. 2017. Scene graph generation by iterative message passing. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5410–5419.

Dongil Yang, Minjin Kim, Sunghwan Kim, Beong-woo Kwak, Minjun Park, Jinseok Hong, Woontack Woo, and Jinyoung Yeo. 2025. Llm meets scene graph: Can large language models understand and generate scene graphs? a benchmark and empirical study. *arXiv preprint arXiv:2505.19510*.

Qifan Yu, Juncheng Li, Longhui Wei, Liang Pang, Wentao Ye, Bosheng Qin, Siliang Tang, Qi Tian, and Yueting Zhuang. 2024. Hallucidoctor: Mitigating hallucinatory toxicity in visual instruction data. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12944–12953.

Rowan Zellers, Mark Yatskar, Sam Thomson, and Yejin Choi. 2018. Neural motifs: Scene graph parsing with global context. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5831–5840.

Zequn Zeng, Yan Xie, Hao Zhang, Chiyu Chen, Bo Chen, and Zhengjue Wang. 2024. Meacap: Memory-augmented zero-shot image captioning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 14100–14110.

Ji Zhang, Kevin J Shih, Ahmed Elgammal, Andrew Tao, and Bryan Catanzaro. 2019. Graphical contrastive losses for scene graph parsing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11535–11543.

Yiwu Zhong, Liwei Wang, Jianshu Chen, Dong Yu, and Yin Li. 2020. Comprehensive image captioning via scene graph decomposition. In *European Conference on Computer Vision*, pages 211–229. Springer.

Terry Yue Zhuo, Zhuang Li, Yujin Huang, Fatemeh Shiri, Weiqing Wang, Gholamreza Haffari, and Yuan-Fang Li. 2023. On robustness of prompt-based semantic parsing with large pre-trained language model: An empirical study on codex. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 1090–1102.

Appendix

A Parsing Evaluation Details

A.1 Complete Baseline Descriptions

We give detailed descriptions of every baseline used in our comparisons.

Sentence Parsing & Merging Methods. These methods first parse each sentence in a multi-sentence caption with a sentence-level parser, then merge the resulting graphs by concatenating triples and removing duplicates. The parsers include Stanford-Parser (Schuster et al., 2015), which converts dependency parses into scene graphs and underlies the original SPICE implementation (Anderson et al., 2016), as well as VG-T5-base (Sent+Merge) and FACTUAL-T5-base (Sent+Merge). The latter two use VG-T5-base (Sharifzadeh et al., 2022) and FACTUAL-T5-base (Li et al., 2023a), respectively, both of which are Flan-T5-base models fine-tuned on the VG and FACTUAL datasets, respectively.

End-to-End (Sentence) Methods. VG-T5-base (Direct) and FACTUAL-T5 (Direct) apply VG-T5-base and FACTUAL-T5-base directly to full captions without using the sentence-level parsing and graph merging pipeline.

End-to-End (Discourse) Methods. These models are fine-tuned on caption-graph pairs from DiscoSG-DS. DiscoSG-T5 variants are Flan-T5 models in base (0.25B), large (0.78B), and xl (3B) sizes, while DiscoSG-Qwen2.5 variants are Qwen2.5 models with 0.5B, 1.5B, and 7B parameters. Unless noted, all are trained on 8,730 examples combining human and synthetic data. We also include DiscoSG-T5-large (300), trained on 300 expert curated examples to control for training size, and DiscoSG-GPT-4o, a GPT-4o teacher fine-tuned on the same 300 examples used to generate the 8,730 synthetic instances.

Few-Shot Prompting (3 shot). We evaluate large instruction tuned PLMs, including Qwen2.5-72B-Instruct, Llama-3.3-70B-Instruct (Grattafiori et al., 2024), and GPT-4o (text-only and multimodal). For each test caption, generation is conditioned on three in-context examples formatted as caption-scene graph pairs; the examples are retrieved from DiscoSG-DS using cosine similarity of TF-IDF vectors. We also evaluate a GraphRAG variant with GPT-4o (text) that first extracts entities and relations from captions and then conditions graph generation on these cues, using 3-shot prompts where each shot provides the caption, its extracted entities and relations, and the reference graph.

Iterative Refinement Methods. All iterative baselines use 3-shot GPT-4o as Generator and Interpreter, differing only in the Programmer component. The three examples for the Generator are retrieved from DiscoSG-DS and those for the Interpreter are retrieved from DiscoSG-ED using the same TF-IDF cosine similarity retrieval method described above.

- **PiVe** (Han et al., 2024): The Programmer is a Flan-T5 model fine-tuned on DiscoSG-ED that proposes insertion actions only.
- **Self-Refine** (Madaan et al.): The Programmer is GPT-4o that generates self-feedback containing insertion and deletion suggestions based on three feedback examples retrieved from DiscoSG-ED using cosine similarities between TF-IDF vectors of captions.
- **Prog-Refine** (Li et al., 2024): The Programmer is a Flan-T5 model fine-tuned on DiscoSG-ED that requires the decoder to directly predict both insertion and deletion actions, unlike other methods that use the encoder for deletion and decoder for insertion.

All methods run **two refinement iterations** for a fair comparison with our framework.

A.2 Metric Definitions

SPICE (Anderson et al., 2016) evaluates the parsing performance by computing an F1 score for each caption based on the overlap between predicted triples (T_{pred}) and gold standard triples (T_{gold}). To better capture semantic equivalence, it uses both exact string and synonym matching for entity nodes within triples. The score is the harmonic mean of precision (P) and recall (R), defined as:

$$\text{SPICE} = \frac{2 \cdot P \cdot R}{P + R}, \quad \text{where:}$$

$$P = \frac{|T_{\text{pred}} \cap T_{\text{gold}}|}{|T_{\text{pred}}|}$$

$$R = \frac{|T_{\text{pred}} \cap T_{\text{gold}}|}{|T_{\text{gold}}|}$$

Bi-SoftSPICE (BSSPICE) is a symmetric version of SoftSPICE (Li et al., 2023a) that replaces exact matching with semantic similarity. SoftSPICE

computes cosine similarities between Sentence-BERT embeddings (Reimers and Gurevych, 2019) of triple phrases from the predicted (G_{pred}) and gold (G_{gold}) graphs, then aggregates these triple similarities into a final graph-level score.

BSSPICE is the harmonic mean of the forward (S_{pg}) and backward (S_{gp}) SoftSPICE scores:

$$S_{\text{pg}} = \text{SoftSPICE}(G_{\text{pred}}, G_{\text{gold}})$$
$$S_{\text{gp}} = \text{SoftSPICE}(G_{\text{gold}}, G_{\text{pred}})$$
$$\text{BSSPICE} = \frac{2 \cdot S_{\text{pg}} \cdot S_{\text{gp}}}{S_{\text{pg}} + S_{\text{gp}}}$$

Inference Time (s) is reported as the average runtime per sample, computed over 100 test captions. For open-source PLMs with fewer than 70 billion parameters, inference is performed on a single NVIDIA RTX 4090 GPU. For API-based models, the runtime is measured using OpenAI's endpoints for GPT-4o-based models and Alibaba's public endpoints for Llama-3.3-70B-Instruct and Qwen2.5-72B-Instruct.

Although DiscoSG-GPT-4o and GPT-4o (text-only) are both based on GPT-4o, DiscoSG-GPT-4o has approximately twice the inference time of GPT-4o (text-only). This is consistent with OpenAI's documentation, which notes that fine-tuned models often experience increased latency due to being hosted on separate infrastructure. As we do not have access to the backend deployment configurations of these proprietary APIs, our measurements serve as an approximate but practical basis for comparing inference efficiency.

A.3 Discourse Error Analysis Methodology

Error Type Taxonomy. To systematically analyse the outputs of discourse-level scene graph parsers, we developed a structured error analysis methodology using GPT-4o as an automated annotator and a human annotator. GPT-4o is guided by a detailed prompt (see Appendix 12). The GPT-4o and human annotators evaluate candidate scene graphs against ground-truth graphs and their source captions. The analysis focuses on four principal categories of discourse-level errors:

- **Cross-Sentence Coreference Resolution:** Failure to correctly resolve anaphoric references, where an entity is mentioned across multiple sentences.
  Example: Not linking "a woman" in one sentence to "she" in a subsequent sentence.
- **Long-Range Relational Dependency:** Omission or misidentification of relationships between entities that span different sentences.
  Example: A connection between two objects mentioned in separate sentences is missing from the graph.
- **Implicit Information Inference:** Failure to find relationships or attributes that are logically implied by the combined context of multiple sentences but not explicitly stated.
  Example: Not inferring the triple (cat, near, window) from the explicit statements (cat, on, mat) and (mat, under, window).
- **Graph Coherence:** The generation of a structurally flawed graph that is fragmented, inconsistent, or incomplete at the discourse level.
  Example: The final graph consists of disconnected subgraphs or omits entities and relations crucial for a unified representation.

For our quantitative analysis, we calculate the prevalence of each error type across the dataset. Each graph is evaluated for the presence or absence of each error type. If one or more instances of a specific error type occur within a graph, that graph is marked as exhibiting that error type and the count for that error type is incremented by one. We then compute the percentage of graphs in the dataset that exhibit each error type.

Analysis. Our results reveal a consistent performance ranking across both automated and human evaluation methods. The GPT-4o-based annotations separate the models into two clear performance tiers. The top tier comprises GPT-4o (text-only), DiscoSG-GPT-4o as the teacher model, and DiscoSG-Refiner-large, all of which achieve comparable average error rates in the range of 66% to 70%. The second tier includes FACTUAL-T5-base (Sent+Merge), which has an average error rate of 82.0%, and DiscoSG-T5-base, which reaches 90.75%. This reflects a performance gap of more than 12 percentage points between the two groups. The findings suggest that leveraging discourse-level data from DiscoSG-DS enhances parsing quality at the discourse level, although smaller PLMs require specialised architectures to remain competitive. Human annotation further confirmed this performance disparity, with evaluators identifying errors in 92.5% of graphs produced by FACTUAL-T5-base (Sent+Merge) and 100% of graphs generated by DiscoSG-T5-base.

	GPT-4o-Annotated					Human-Annotated				
	Cross.↓	**Long.↓**	**Impl.↓**	**Graph.↓**	**Avg.↓**	**Cross.↓**	**Long.↓**	**Impl.↓**	**Graph.↓**	**Avg.↓**
FACTUAL-T5-base (Sent+Merge)	61.0	97.0	77.0	93.0	82.00	100.0	100.0	70.0	100.0	92.5
DiscoSG-T5-base	67.0	100.0	96.0	100.0	90.75	100.0	100.0	100.0	100.0	100.0
GPT-4o (text-only)	36.0	81.0	83.0	65.0	66.25	60.0	80.0	80.0	90.0	77.5
DiscoSG-GPT-4o (Teacher)	28.0	86.0	76.0	81.0	67.75	50.0	80.0	80.0	100.0	77.5
DiscoSG-Refiner-large	43.0	85.0	74.0	79.0	70.25	30.0	100.0	100.0	80.0	77.5
Average	47.0	89.8	81.2	83.6	75.4	68.0	92.0	86.0	94.0	85.0

Table 7: Discourse parsing error rates (%) for each parsing method, comparing automated error analysis by GPT-4o (**left**) against manual annotation (**right**). Lower values indicate better performance. Error categories are abbreviated as follows: **Cross.** (Cross-Sentence Coreference), **Long.** (Long-Range Dependency), **Impl.** (Implicit Inference), and **Graph.** (Graph Coherence). The automated analysis used the prompt shown in Appendix 12.

To validate our automated evaluation, we measured inter-annotator agreement between GPT-4o and human experts on a subset of 50 samples. The analysis yielded a Jaccard score of 70.9% and an F1 score of 74.4%, indicating substantial agreement and demonstrating that GPT-4o serves as a reliable proxy for human judgment. This validation enables scalable and reproducible assessment of discourse-level phenomena, which would otherwise be infeasible to evaluate manually at scale.

A.4 Implementation Details

We implement all models using the Hugging Face Transformers[3] library and conduct experiments on a single NVIDIA A100 80GB or RTX 4090 GPU. To ensure fair inference time comparisons, all speed measurements for open-source models with fewer than 70 billion parameters are conducted exclusively on the RTX 4090 GPU. Our Programmer models are trained for 3 epochs with a batch size of 2. For the composite loss function, the deletion loss ($\mathcal{L}_{\text{delete}}$) and insertion loss ($\mathcal{L}_{\text{insert}}$) are weighted equally, using coefficients $\lambda_{\text{del}} = \lambda_{\text{ins}} = 0.5$.

Human annotators involved in the manual data labelling process were compensated at a rate commensurate with the average local salary to ensure fair labour practices.

A.5 Case Study

We present a visual example to illustrate the iterative refinement process of the DiscoSG framework. At the top, we show the input image (for reference only) alongside its multi-sentence textual description. The middle section displays the initial scene graph parsed from the description. At the bottom, we show the refined graph after applying the deletion and insertion modules, which remove redundant or irrelevant triples and add new, contextually appropriate ones. This example highlights how our framework enhances the accuracy and coherence of discourse-level scene graphs for complex image descriptions. See Figure 1 for details.

[3] https://huggingface.co/

B Data Collection Details

B.1 Active Learning Annotation for DiscoSG-DS

The pseudocode for our active learning annotation process is detailed in Algorithm 1.

Algorithm 1: Active Learning Pipeline

Input: Seed training set D_{seed} ($|D_{\text{seed}}| = 62$); fixed validation set D_{val} ($|D_{\text{val}}| = 40$); unlabeled data pool U; number of iterations $N_{\text{iter}} = 2$; batch sizes per iteration $S = \{94, 204\}$.

Output: Expanded training set D_{train}; sequence of fine tuned models $\{M_i\}_{i=0}^{N_{\text{iter}}}$, validation scores $\{P_i\}_{i=0}^{N_{\text{iter}}}$.

$D_{\text{train}} \leftarrow D_{\text{seed}}$;
$M_0 \leftarrow \text{FineTune}(\text{GPT-4o}, D_{\text{train}})$;
$P_0 \leftarrow \text{Evaluate}(M_0, D_{\text{val}})$; `/* initial SPICE score */`
for $i \leftarrow 0$ **to** $N_{iter} - 1$ **do**
  $B_{\text{raw}} \leftarrow \text{Sample}(S[i], U)$;
  $U \leftarrow U \setminus B_{\text{raw}}$;
  $B_{\text{draft}} \leftarrow \text{Predict}(M_i, B_{\text{raw}})$;
  $B_{\text{refined}} \leftarrow \text{HumanRefine}(B_{\text{draft}})$; `/* two stage expert refinement */`
  $D_{\text{train}} \leftarrow D_{\text{train}} \cup B_{\text{refined}}$;
  $M_{i+1} \leftarrow \text{FineTune}(\text{GPT-4o}, D_{\text{train}})$;
  $P_{i+1} \leftarrow \text{Evaluate}(M_{i+1}, D_{\text{val}})$;
end
return $D_{train}, \{M_i\}_{i=0}^{N_{iter}}, \{P_i\}_{i=0}^{N_{iter}}$

B.2 Annotation Guidelines for DiscoSG-DS

Annotation was conducted using Amazon Mechanical Turk's sandbox interface. Given the minimal

Image:

Caption:
In the image, a group of people are seen walking on a concrete pier towards a ferry terminal. The pier is equipped with a metal railing on the left side, providing safety for the pedestrians. The individuals are casually dressed, suitable for a summer day, and are carrying various bags and backpacks, suggesting they might be travelers or commuters. Two red flags are flying on the left side of the image, possibly indicating some sort of warning or information. Further ahead, a blue and white canopy can be seen, likely providing shelter at the ferry terminal. The sky above is hazy, creating a serene atmosphere. In the distance, tall buildings loom, indicating that the location is near a city or urban area. The image does not contain any discernible text. The relative positions of the objects suggest a typical scene at a ferry terminal: people moving towards their destination, the safety measures in place, and the urban backdrop adding context to the setting. The image captures a moment of everyday life, with each element playing its part in the narrative.

Initial Graph:
(people , walk on , pier) , (people , walk towards , ferry terminal) , (pier , is , concrete) , (railing , at the left of , pier) , (railing , is , metal) , (individuals , carry , backpacks) , (individuals , carry , bags) , (individuals , is , casually dressed) , (flags , fly at the left of , image) , (flags , is , 2) , (flags , is , red) , (canopy , at , ferry terminal) , (canopy , is , blue) , (canopy , is , white) , (sky , is , hazy) , (sky , is , serene) , (buildings , is , tall) , (buildings , near , city) , (image) , (backdrop , is , urban) , (objects , at , terminal) , (objects , in , setting) , (objects , is , relative) , (people , move towards , destination) , (terminal , is , ferry) , (element , play in , narrative) , (image , capture , narrative) , (narrative , is , everyday)

Deletion Prediction:
(people , walk on , pier) , (people , walk towards , ferry terminal) , (pier , is , concrete) , (railing , at the left of , pier) , (railing , is , metal) , (individuals , carry , backpacks) , (individuals , carry , bags) , (individuals , is , casually dressed) , (flags , fly at the left of , image) , (flags , is , 2) , (flags , is , red) , (canopy , at , ferry terminal) , (canopy , is , blue) , (canopy , is , white) , (sky , is , hazy) , (sky , is , serene) , (buildings , is , tall) , (buildings , near , city) , ~~(image)~~ , (backdrop , is , urban) , ~~(objects , at , terminal) , (objects , in , setting) , (objects , is , relative) , (people , move towards , destination) , (terminal , is , ferry) , (element , play in , narrative) , (image , capture , narrative) , (narrative , is , everyday)~~

Insertion Generation:
(railing , at the left of , pier) , (sky , is , hazy) , (people , walk on , pier) , (flags , is , 2) , (individuals , carry , backpacks) , (buildings , near , ferry terminal) , (pier , is , concrete) , (railing , is , metal) , (buildings , near , city) , (backdrop , is , urban) , (individuals , is , casually dressed) , (individuals , carry , bags) , (sky , is , serene) , (flags , fly at the left of , image) , (people , walk towards , ferry terminal) , (canopy , is , white) , (canopy , is , blue) , (canopy , at , ferry terminal) , (flags , is , red) , (buildings , is , tall) , (people , is , group of)

Figure 1: Visualisation of the iterative scene graph refinement process in the DiscoSG-Refiner framework.

risk associated with scene graph annotation tasks (extracting object relationships from visual descriptions), no additional risk disclaimers were required beyond standard research participation protocols.

To ensure the quality and consistency of our dataset, we established comprehensive guidelines for annotating scene graphs ($\boldsymbol{y}$) from discourse descriptions ($\boldsymbol{x}$) and images. The key principles and procedures are summarised below.

Core Task & Principles. The primary objective is to extract a scene graph grounded in both the text and the image. This involves identifying object-attribute-value triples, such as `(mast, is, wooden)`, and object-relation-object triples, like `(mast, on, deck)`. All relations derived from verbs are explicitly marked with a `v:` prefix (e.g., `v:rest on`). The `v:` prefix is only applied during annotation to align with the FACTUAL-MR format (Li et al., 2023a) and is later removed. Only objects clearly visible or unambiguously inferable from the image are included as nodes. Abstract concepts (e.g., "game"), viewpoint references ("camera"), and non-localizable elements are excluded.

Annotation Workflow and Quality Control. The annotation process follows a structured workflow to maximise factual accuracy and consistency.

- **Hallucination Handling:** Annotators first cross-reference the text description ($\boldsymbol{x}$) with the image to detect factual inconsistencies (hallucinations), such as incorrect object counts or non-existent relations. Instances with significant hallucinations that compromise core scene understanding are flagged and typically excluded from full annotation. Minor descriptive errors are corrected by the annotators.
- **Refinement Process:** Annotation is performed using a dedicated interface on Amazon Mechanical Turk[4] where annotators refine initial draft graphs generated by FACTUAL-T5-base (Sent+Merge) or DiscoSG-GPT-4o. This involves adding, deleting, or modifying nodes (entities) and edges (relations).
- **Expert Review:** All manual annotations undergo a two-stage expert review that requires consensus for finalisation, ensuring high-quality outputs.
- **Intermediate Representation:** To reduce ambiguity during annotation, we use the FACTUAL-MR format (Li et al., 2023a), which is then programmatically converted into the final scene graph structure.

Semantic Representation Rules. Specific rules are applied to the representation of object attributes and relations:

- **Attributes:** Object attributes are represented using either the explicit form `(object, has_attribute, attribute_value)` or the shorthand form `(object, is, attribute_value)`.
- **Relations:** Relations between objects or nodes follow the FACTUAL-MR convention, consisting of verbs and prepositions, with a preference for active voice where applicable. For example, `(court, v:separate by, lines)` is rewritten as `(lines, v:separate, court)` to ensure consistency, following the annotation guidelines of FACTUAL (Li et al., 2023a).

Handling Discourse Phenomena. To address key discourse-level challenges such as Cross-Sentence Coreference, Long-Range Relational Dependencies, Implicit Inference, and Graph Coherence, our annotation guidelines emphasise the following procedures:

- **Entity Disambiguation and Coreference:** To resolve ambiguity, annotators create unique nodes for each distinct object, even if they share the same name. For instance, in "Two cats are visible: one is sleeping, the other is playing," the cats are annotated as separate nodes (e.g., `cat:1`, `cat:2`). Similarly, collective nouns like "dogs" are disaggregated into individual nodes when described with different attributes or relations. For example, the phrase "There are two dogs, one brown and one white" results in distinct triples `(dog:1, is, brown)` and `(dog:2, is, white)`.
- **Quantifier Annotation:** Annotators capture the quantity of objects mentioned in the text (e.g., "2 people", "several cats") using a predefined set of labels (e.g., `1`, `2`, `many`, `uncountable`) associated with the corresponding entity nodes. To handle discourse-level counts, annotators aggregate these quantifiers across sentences. For instance, if one sentence mentions "a cat" and a later sentence mentions "another cat," the final count for the "cat" entity is consolidated to the quantifier label `2`.

[4] https://www.mturk.com/

- **Implicit Relation Inference:** Annotators enrich the graph by inferring and adding relations that are logically implied by the context but not explicitly stated in a single sentence. For example, given the statements `(cat, on, mat)` and `(mat, under, window)`, the implicit spatial relation `(cat, near, window)` is added to the graph.
- **Entity Specificity and Semantic Precision:** Annotators are guided to use the most specific entity representation available (e.g., preferring "cat" over "animal") and to resolve semantic redundancy by merging nodes where appropriate (e.g., representing "a husband and a wife" as a single "couple" node).

B.3 Construction of the D-FOIL Dataset

The D-FOIL dataset was constructed through a multi-stage process designed to create a corpus of hallucinated captions, their corrected counterparts with minimal editing, and factually grounded reference captions.

1. **Initial Data Collection:** We first gathered 410 image-text pairs where the captions were identified as containing discourse-level hallucinations. We also annotated the locations of these hallucinated entities or relations.
2. **Reference Generation:** To establish a hallucination-free ground truth for each image, we employed GPT-4.1 to generate a new, detailed description. To maintain consistency with existing datasets, we used the same prompt as the SharedGPT4V dataset:

   Reference Generation Prompt

   Create detailed captions describing the contents of the given image. Include the object types and colours, counting the objects, object actions, precise object locations, texts, double-checking relative positions between objects, etc.

   These generated descriptions served as the factual references for the subsequent correction step.
3. **Human Correction:** Using the generated references as a guide, our annotators manually revised and corrected a subset of 200 of the original hallucinated captions. This step ensured that the final corrected captions were factually aligned with the image content.

Datasets	Captions		Graph	
	MATTR ↑	MTLD ↑	MATTR ↑	MTLD ↑
VG	0.3682	13.5043	0.3563	12.5834
FACTUAL	0.3536	13.1511	0.3549	12.3893
TSGBench	0.5168	9.8000	0.5137	9.2154
DiscoSG-DS	0.3809	14.1317	0.3596	13.1562
DiscoSG-ED *15*×	0.3809	14.1317	0.3603	12.4330

Table 8: Evaluation of lexical diversity in captions and scene graphs across different datasets.

Each finalised instance in the D-FOIL dataset comprises three key components: 1) the original caption containing hallucinations, 2) the corrected hallucination-free version of the hallucinated caption, and 3) the hallucination-free reference description generated by GPT-4.1.

B.4 Annotator Recruitment and Compensation

Annotators were recruited from the research team based on their specialised expertise in scene graph parsing. All annotators were compensated through their regular institutional salaries with no additional per-annotation payments, ensuring fair labour practices. The recruitment of internal team members was necessary due to the specialised knowledge required for discourse-level scene graph annotation. To mitigate potential bias from this arrangement, we implemented rigorous quality control measures, including inter-annotator agreement assessment and two-stage expert review.

B.5 Dataset Licensing

The DiscoSG-DS and D-FOIL datasets will be released under [MIT/Apache 2.0/CC BY 4.0] license for research purposes. The dataset includes expert annotations and synthesised examples that build upon existing research datasets, with appropriate consent obtained from research team members who participated in annotation. The dataset is intended for academic research use only and builds upon publicly available research datasets where consent was handled during original collection.

C Lexical Diversity Evaluation

To quantify linguistic diversity, we evaluate both the caption text and the linearised graph tokens of each dataset using two length-insensitive measures derived from the type-token ratio (TTR), the proportion of unique words (types) to total words (tokens). Because raw TTR declines with increasing

text length, we adopt two widely accepted alternatives that offer more stable estimates:

- **Moving-Average TTR (MATTR)** (Covington and McFall, 2010) : computes the average TTR within a sliding window, reducing sensitivity to text length.
- **Measure of Textual Lexical Diversity (MTLD)** (McCarthy, 2005) : measures the average number of tokens needed before the running TTR drops below a threshold, with higher scores indicating greater vocabulary diversity and lower redundancy.

To reduce the impact of varying average caption and graph token lengths across datasets, we compute diversity scores at the corpus level by concatenating all captions and all graph tokens, respectively, into single text sequences before computing diversity scores. This approach ensures a fair comparison across datasets. For both metrics, higher scores indicate greater lexical variety. Results are shown in Table 8.

Analysis. Table 8 shows that DiscoSG-DS achieves the highest MTLD scores, with 14.13 for captions and 13.16 for graph tokens, reflecting its strongest long-range lexical diversity among all datasets. While TSGBench records the highest MATTR values, at 0.517 for captions and 0.514 for graphs, its much lower MTLD scores, 9.80 for captions and 9.22 for graphs, indicate that its lexical variety is not sustained over longer sequences. In contrast, DiscoSG-DS maintains a strong balance of both local and global diversity, making it particularly suitable for discourse-level modelling. The synthetic extension, DiscoSG-ED, preserves this lexical richness, demonstrating that data augmentation does not diminish linguistic diversity.

D Prompt Descriptions

Table 9 provides an overview of the prompt templates and example inputs used in our experiments. These templates standardize input formats and task-specific instructions across diverse scene graph parsing and caption analysis tasks, ensuring consistency in model prompting and evaluation.

Template Name	Description	Link
DiscoSG-Refiner Deletion Prediction Template	Provides instructions for identifying and removing incorrect or irrelevant triplets from a candidate scene graph, based on a given caption.	Figure 2
DiscoSG-Refiner Deletion Prediction Example	Shows a concrete input example used in the Deletion Prediction task.	Figure 3
DiscoSG-Refiner Insertion Generation Template	Guides the model to add missing but contextually appropriate triplets to an incomplete scene graph, based on the caption and current graph.	Figure 4
DiscoSG-Refiner Insertion Generation Example	Provides an example input for the Insertion Generation task.	Figure 5
Hallucination Detection Template	Instructs the vision-language model to detect hallucinated content by comparing a caption to image content, producing a binary decision. This is used to filter data during synthetic data collection for DiscoSG-DS.	Figure 6
PiVe Prompt Template	Guides GPT-4o to generate initial scene graphs via few-shot prompting, then refine them using feedback from the DiscoSG-Refiner Programmer, conditioned on the caption.	Figure 7
D-FOIL Hallucination Detection Template	Prompts GPT-4o to determine which of two candidate captions is more semantically aligned with a reference caption, helping detect hallucinations.	Figure 8
Self-Refine Feedback Generation Template	Instructs GPT-4o to generate self-feedback on a scene graph, suggesting triplet insertions or deletions.	Figure 9
Self-Refine Refinement Template	Prompts GPT-4o to revise a previous scene graph based on generated self-feedback.	Figure 10
Prog-Refine Template	Applies a sequence of edits—insertions and deletions—predicted by DiscoSG-Refiner to update a scene graph.	Figure 11
Discourse-Level Error Analysis Template	Prompts the model to analyze and classify scene graph errors by comparing with ground truth, focusing on coreference, relational dependencies, implicit inference, and coherence.	Figure 12

Table 9: Descriptions of the prompt templates used in our experiments.

Delete Task:
Caption:
{Caption}
Candidate Graph:
{Graph needs refine}

Figure 2: Prompt template for deletion prediction in DiscoSG-Refiner

Delete Task:
Caption: The image captures a serene scene in a park. A gravel path, dappled with sunlight filtering through the tall trees on either side, winds its way towards a white bridge. The bridge arches over a small body of water, possibly a stream or a pond. The sky above is a clear blue, with a few clouds scattered across it. The predominant colors in the image are the lush greens of the trees and grass, and the blue of the sky. The perspective of the image follows the path, leading the viewer's eye towards the bridge in the distance. The image exudes a sense of tranquility and invites one to take a leisurely stroll down the path.
Candidate Graph: (path , is , gravel) , (trees , is , lush) , (trees , is , tall) , (path , wind towards , bridge) , (bridge , is , white) , (sky , is , gray) , (bridge , arch over , water) , (sky , is , clear) , (sky , is , blue) , (clouds , scatter across , sky) , (grass , is , lush) , (trees , on either side of , path) , (water , is , small) , (sunlight , dapple , path) , (sunlight , filter through , trees)

Figure 3: Example input prompt for deletion prediction in DiscoSG-Refiner

Insert Task:
Caption:
{Caption}
Corrupted Graph:
{Graph needs refine / Graph refined by deletion prediction task}

Figure 4: Template input prompt for insertion generation in DiscoSG-Refiner

Insert Task:
Caption: The image captures a serene scene in a park. A gravel path, dappled with sunlight filtering through the tall trees on either side, winds its way towards a white bridge. The bridge arches over a small body of water, possibly a stream or a pond. The sky above is a clear blue, with a few clouds scattered across it. The predominant colors in the image are the lush greens of the trees and grass, and the blue of the sky. The perspective of the image follows the path, leading the viewer's eye towards the bridge in the distance. The image exudes a sense of tranquility and invites one to take a leisurely stroll down the path.
Corrupted Graph: (path , is , gravel) , (trees , is , lush) , (trees , is , tall) , (path , wind towards , bridge) , (bridge , is , white) , ~~"(sky , is , gray)~~Deleted by Deletion Prediction" , (bridge , arch over , water) , (sky , is , clear) , (sky , is , blue) , (clouds , scatter across , sky) , (grass , is , lush) , (trees , on either side of , path) , (water , is , small) , (sunlight , dapple , path) , (sunlight , filter through , trees)

Figure 5: Example prompt for the insertion-generation step in DiscoSG-Refiner

You are given an image and its description.

Analyse the description in detail to determine if it includes any hallucinations (information not present in the image).

Provide a detailed explanation of your reasoning.

At the end, provide a binary decision in the format:

"Final Decision: [Yes/No]".

Description:
{description}
{image upload here}

Figure 6: Prompt template for detecting hallucinations by comparing a caption to its corresponding image.

Transform the following text into a complete semantic graph and add the provided triple to the generated semantic graph.

Example 1:

Caption: {Caption}
Incomplete semantic graph: {incomplete scene graph}
Triple to add: {triple to add}
Complete semantic graph: {complete graph}

Example 2:

Caption: {Caption}
Incomplete semantic graph: {incomplete scene graph}
Triple to add: {triple to add}
Complete semantic graph: {complete graph}

Example 3:

Caption: {Caption}
Incomplete semantic graph: {incomplete scene graph}
Triple to add: {triple to add}
Complete semantic graph: {complete graph}

Now, transform the following caption into a complete semantic graph:

Caption: caption

Incomplete semantic graph:
{incomplete graph}

Triple to add:
{triple need to add} # Generated by our Refiner

Complete semantic graph:

Figure 7: Prompt template used by PiVe to complete a scene graph by adding a specified triplet.

You are given a ground truth image caption and two candidate captions. Your task is to choose which candidate caption (Candidate 1 or Candidate 2) is closer in meaning and detail to the ground truth caption. Only output "Candidate 1" or "Candidate 2". Do not provide any explanation or analysis.

Ground truth caption: {ground truth}
Candidate 1: {cand1}
Candidate 2: {cand2}

Which candidate is closer to the ground truth caption?

Figure 8: Prompt template for identifying hallucinations in candidate captions by selecting the one that best aligns with the ground truth in D-FOIL.

You are an expert at analyzing scene graphs for accuracy and completeness. Your task is to evaluate a scene graph based on an image caption and suggest improvements.

A scene graph consists of triples in the format (subject, relation, object) that represent the entities and relationships in an image.

For the given caption and scene graph, identify:

1. Triples that need to be added (entities or relationships mentioned in the caption but missing from the graph)

2. Triples that need to be removed (incorrect, irrelevant, or redundant entries) Here are some examples of how to provide refinement suggestions:

Example 1:

Caption: {caption}
Scene Graph: {scene graph}
insert triples: {triples need to insert}
delete triples: {triples need to delete}

Example 2:

Caption: {caption}
Scene Graph: {scene graph}
insert triples: {triples need to insert}
delete triples: {triples need to delete}

Example 3:

Caption: {caption}
Scene Graph: {scene graph}
insert triples: {triples need to insert}
delete triples: {triples need to delete}

—

Now, please analyze this new case:
Caption: {caption}
Scene Graph: {scene graph}

Provide your recommendations in this format:
insert triples: {All triples that should be added}
delete triples: {All triples that should be removed}

Figure 9: Prompt template for Self-Refine that guides GPT-4o to generate feedback by identifying insertions and deletions needed to improve a scene graph based on a caption.

Parse the following image description into a scene graph. A scene graph consists of triplets in the format (subject, relation, object).

I'll provide examples that show how to refine scene graphs based on feedback suggestions:

Example 1:

Caption: {caption}
Corrupted Graph: {Corrupted scene graph}
Refinement Suggestion:
insert triples: {all triples that should be added}
delete triples: {all triples that should be removed}
Improved Scene Graph: {Target scene graph}

Example 2:

Caption: {caption}
Corrupted Graph: {Corrupted scene graph}
Refinement Suggestion:
insert triples: {all triples that should be added}
delete triples: {all triples that should be removed}
Improved Scene Graph: {Target scene graph}

Example 3:

Caption: {caption}
Corrupted Graph: {Corrupted scene graph}
Refinement Suggestion:
insert triples: {all triples that should be added}
delete triples: {all triples that should be removed}
Improved Scene Graph: {Target scene graph}

Now, generate an accurate scene graph for the following description:

Description:
{input_caption}

Previous Graph:
{previous_graph}

Refinement Suggestion:
{Self-Feedback from Self-Refine}

Generate an improved scene graph below:

Figure 10: Prompt template used in Self-Refine for revising a scene graph according to GPT-4o-generated feedback.

You are an expert at improving scene graphs based on edit actions. A scene graph consists of triples in the format (subject, relation, object).

Here are some examples of how to improve scene graphs based on edit actions:

Example 1:

Caption: {caption}
Scene Graph: {scene graph}
Edit Actions:
INSERT: {target triples need to insert}
DELETE: {target triples need to delete}
Improved Scene Graph: {original scene graph}

Example 2:

Caption: {caption}
Scene Graph: {scene graph}
Edit Actions:
INSERT: {target triples need to insert}
DELETE: {target triples need to delete}
Improved Scene Graph: {original scene graph}

Example 3:

Caption: {caption}
Scene Graph: {scene graph}
Edit Actions:
INSERT: {target triples need to insert}
DELETE: {target triples need to delete}
Improved Scene Graph: {original scene graph}

—
Now, please improve this new scene graph:

Caption: {caption}

Scene Graph: {scene graph}

Edit Actions: {programmer output}

Improved Scene Graph:

Figure 11: Prompt template used in Prog-Refine for applying edit actions to revise a scene graph.

Given the following inputs:
1. caption: A textual description of a scene
2. ground truth graph: The correct reference scene graph
3. candidate graph: The scene graph being evaluated

Analyze the scene graph parsing error by comparing the candidate graph against the ground truth graph and the caption. Determine which of the following error types best describes the primary issue. **For each error type, consider both errors of commission (incorrect outputs) and errors of omission (missing outputs that should be present). Pay special attention to cases where the candidate graph fails to produce entities, links, or relations that are present in the ground truth graph or are required by the caption.**

1. Cross-Sentence Coreference Resolution Error
Definition: Failure to correctly identify and link mentions of the same entity across different sentences, including both incorrect links and entirely missing coreference chains.
- Omission focus: Also count cases where coreference chains are completely absent because the parser failed to produce the necessary entities or links.
- Example: If "a woman" is mentioned in one sentence and later referred to as "she" or "the artist" in another, but the candidate graph either mislinks or fails to link these as the same entity, or does not produce the entities at all.
2. Long-Range Relational Dependency Error
Definition: Missing or incorrect relationships between entities mentioned in separate, potentially distant sentences.
- Omission focus: Count not only incorrect relations, but also missing long-range dependencies—i.e., when the candidate graph produces very few or no relations that should span across sentences.
- Example: If an object is introduced in one sentence and its relationship to another object is described in a different sentence, but the candidate graph fails to connect these entities, or omits the relationship entirely.
3. Implicit Information Inference Error
Definition: Failure to infer and represent relationships or attributes not explicitly stated but apparent from broader context.
- Omission focus: This category is inherently about omissions—specifically, failure to cover information that should be inferred based on the combined information in the caption and ground truth graph.
- Example: If the ground truth graph contains a relationship or attribute not textually explicit in the caption (e.g., inferring (cat, near, window) from "The cat is on the mat." and "The mat is under the window."), but the candidate graph omits this, it is an omission error.
4. Graph Coherence Error
Definition: Failures in producing a globally consistent representation of the entire scene, including contradictions, fragmentation, or incompleteness.
- Omission focus: Include cases of "graph incompleteness"—that is, when the overall graph lacks sufficient entities, links, or triplets to represent the complete scene described in the text.
- Example: The candidate graph only contains disconnected subgraphs or lacks key triplets, failing to provide a unified and comprehensive scene graph.
5. Others
Any error type that doesn't fit into the categories above.
—
For each error category that applies, do the following:
- State the category name as a section heading.
- Only consider the differences between the candidate graph and the ground truth graph, and between the candidate graph and the caption, strictly based on the provided data. Do not incorporate any information or assumptions beyond what is explicitly given in the data.
- Provide detailed reasoning, including:
- Specific evidence from the graph (e.g., which triples or nodes are missing or illustrate the error).
- An explanation of why this constitutes the given error type (refer to the definitions and examples above, including omission criteria).
- An example or clarification if helpful.
—
At the end, under the heading "Applicable Error Categories", list all applicable error category labels (using the exact English names, separated by commas).

Inputs:
Caption:
{caption}
Ground Truth Graph:
{ground truth graph}
Candidate Graph:
{candidate graph}

Figure 12: Prompt template for analysing discourse-level scene graph errors, including coreference, long-range relations, implicit inference, and coherence, by comparing candidate graphs with ground truth graphs and captions.