Expressive scene graph generation for rich visual understanding: Overcoming limitations in evaluation, datasets, and neural network architectures

Julian Lorenz

Angaben zur Veröffentlichung / Publication details:

Lorenz, Julian. 2026. "Expressive scene graph generation for rich visual understanding: Overcoming limitations in evaluation, datasets, and neural network architectures." Augsburg: Universität Augsburg.

Dissertation

Expressive Scene Graph Generation for Rich Visual Understanding

Overcoming Limitations in Evaluation, Datasets, and Neural Network Architectures

Julian Lorenz
aus Hannover

Eingereicht am
17. Dezember 2025

an der
Fakultät für Angewandte Informatik
Universität Augsburg

zur Erlangung des Doktorgrades
Dr. rer. nat.

Fakultät für Angewandte Informatik
Universität Augsburg

Gutachter:	Prof. Dr. rer. nat. Rainer Lienhart Prof. Dr. phil. Annemarie Friedrich Prof. Dr.-Ing. Eckehard Steinbach
Tag der Verteidigung:	23. Februar 2026

Abstract

Scene graph generation (SGG) is the task of representing a visual scene image as a graph of subjects and objects and their relationships with each other. In a scene graph, the subjects and objects in the scene are represented by the graph's vertices while the relationships are represented by the edges. A scene graph can capture rich interactions between objects and people and encode this information in an efficient data structure, ready to be processed by downstream applications.

This thesis addresses two main challenges of current work on scene graph generation, namely limitations of existing evaluation procedures and limited expressiveness and reliability of existing scene graph datasets. We introduce a detailed overview of commonly used SGG metrics and provide an efficient reference implementation called *SGBench* which is independent of any deep learning frameworks. Furthermore, we investigate recent publications on one-stage SGG methods that claim state-of-the-art performance. Our analysis reveals that prior claims are based on a flawed evaluation, which skews the reported scores in favor of newer models. As a response, we rigorously highlight the differences in the applied evaluation protocols and report the actual scores under a fair comparison. Additionally, we propose DSFormer, a novel two-stage SGG architecture which proves that - contrary to recent trends - two-stage methods can actually outperform one-stage methods. Since most scene graph datasets severely underrepresent rare predicate classes, we introduce Haystack, a new dataset that provides reliable annotations and enables the application of a wider range of metrics to scene graph generation. These new metrics reveal new properties about existing SGG methods. Finally, we present CoPa-SG, a large synthetic scene graph dataset with over 86M relation annotations. This dataset introduces two new relation types, parametric relations and proto-relations which capture more fine-grained and informative details about the environment.

By addressing critical limitations in evaluation, datasets, and architectural design, this thesis contributes to the advancement of scene graph generation, paving the way for SGG methods capable of capturing the richness and complexity of visual scenes.

Acknowledgments

Researching at the Machine Learning and Computer Vision Lab has been a wonderful and very rewarding experience, one that I look back at with much gratitude.

I am grateful to Prof. Rainer Lienhart, who supported me throughout my time at his lab. The discussions we had about ongoing research sparked new ideas, helped me move forward, and significantly shaped the direction of my work. I appreciate the trust that he put into me, encouraging me to pursue the many research ideas I had while working at his lab. I am also grateful to Prof. Annemarie Friedrich and Prof. Eckehard Steinbach for their time and effort they put into reviewing this dissertation.

During my time as a PhD candidate, I had the opportunity to work with many wonderful colleagues, who supported me in many different ways. I am thankful to Katja Ludwig for her mentorship and valuable advice that helped me a lot to get started. Robin Schön's breadth of knowledge were invaluable, and I have always appreciated his opinion on recent developments in research. Discussions with Daniel Kienzle inspired many successful approaches and often provided breakthroughs when facing challenges. I am thankful to Florian Barthel, who, during his short time at the lab, assisted with getting started on my first research project and really got me motivated to dive into this exciting research field. I am grateful to all my past and present colleagues. Together we have created a supportive atmosphere and great community that you could always rely on.

Finally, I am grateful to my family for their unwavering support and encouragement. Their kindness and understanding were essential in enabling me to complete my dissertation.

Contents

Acronyms

PredCls	Predicate Classification. 19, 20, 45–47, 49, 50, 52, 71, 73, 105, 106, 111
PSGG	Panoptic Scene Graph Generation. 3, 10, 33, 35–37, 46, 49, 56
SGCls	Scene Graph Classification. 19, 20, 105
SGDet	Scene Graph Detection. 18–20, 46, 47, 49, 50, 56, 105, 106
SGG	Scene Graph Generation. iii, 2–4, 6, 7, 9–13, 15, 17, 18, 21, 23, 26, 27, 29, 30, 32, 35–40, 45, 46, 50, 53, 56, 57, 59, 61, 62, 66, 67, 74, 79–81, 83, 86, 89–92, 94, 101–103, 105–107, 111, 112

1 Introduction

Visual scene understanding, the ability for computers to interpret and reason about the content of images, has been a central challenge in computer vision research since the 1960s [37, 108]. The ability to understand a scene and not just identify objects but also to grasp their relationships, context, and potential actions, unlocks a vast range of applications. From enabling autonomous vehicles to navigate safely and reliably, to assisting visually impaired individuals with environmental awareness, to powering more intuitive human-computer interaction, robust scene understanding is a critical step towards truly intelligent assistants.

Since the pioneering research, a lot of progress has been made in this field, especially when neural networks were first introduced for image classification using AlexNet [62]. Many advancements quickly followed, such as object detection [32], object segmentation [118] or visual question answering [4, 50], just to name a few. The introduction of the transformer architecture to natural language processing [127] and computer vision [27] eventually paved the way for a major milestone in scene understanding using large multi-modal foundation models. Trained with enormous computing resources on giant datasets, these vision language models (VLMs) have versatile capabilities to describe and represent visual scenes using natural language. Given an image, VLMs can be queried to describe parts of the scene or even conduct reasoning on it.

Although these large-scale models exhibit impressive capabilities, they require immense resources which makes them impractical to run locally for most actors. Additionally, they are reliant on large-scale training and cannot be easily adapted to niche applications. This is where scene graphs become practical. Originally introduced by Johnson et al. [49] in 2015, scene graphs provide a structural representation of an image as a graph. This graph contains information about the various objects in an image (e.g., people or cars) and how they interact with each other. In a scene graph, the objects in the scene are represented by the graph's vertices while the relationships are represented by the edges. Compared to vision-language data, scene graphs are much easier to train and provide a rich and well-defined data structure that can be reliably processed by other applications to assist in complex scene understanding tasks. Scene graphs are used for numerous tasks like navigation [121], planning [36], reasoning [44, 48], assisting surgery [45, 119], or analyzing sports [42, 102, 132].

In fact, VLMs have recently started to shift back towards more structured training data. Molmo [22], for example, includes annotations in its training dataset that describe where human annotators would point at when reasoning about a specific

scene. This enables the model to produce more reliable information when analyzing an image.

1.1 Problem Definition and Challenges

Scene Graph Generation In computer science in general, the name "Scene Graph" is used for multiple somewhat related domains which must not be confused with each other. In computer graphics, a scene graph is a data structure that is used in computer games and vector-based graphics editors, and describes the logical and spatial representation of a scene [19] in a hierarchy. In robotics [5, 46], scene graphs refer to a hierarchical data structure that can be used for actions like planning or navigation. The graph groups the scene into cameras, objects, rooms, and buildings and encodes the relationships between these entities.

For this thesis, we use the definition for computer vision, introduced by Johnson et al. [49]. They define a scene graph as a collection of objects in a scene (e.g., "car" or "person"), which are connected by relations with each other (e.g., "driving" or "sitting on"). Additionally, scene graphs contain attributes that more closely describe assigned objects (e.g., "the bench is made out of wood"). Most recent work on scene graph generation focuses on detecting the relations in an image while omitting the attributes. We follow this common approach and view scene graphs as a collection of instances and relations. A more technical definition of scene graphs follows in section 2.1.

Evaluating SGG methods is non-trivial and limited As we will cover in section 3.5, scene graphs are non-trivial to evaluate. In fact, in chapter 4, we will discuss several prior works that, knowingly or not, misinterpret the commonly used definition of the employed metric, inflating the reported benchmark scores. Evaluating scene graphs involves various steps, and the evaluation has to be carefully adapted if novel network architectures return slightly different output types.

The usually employed metrics in scene graph generation are a consequence of the available limited scene graph datasets. Especially, sparse annotations in scene graph datasets are a limiting factor. Since only sparse annotations are available, prior scene graph metrics have to fall back to ranking metrics which only quantify how good the SGG model is at retrieving salient relations similar to a human. We argue that the point of a scene graph lies in its dense and reliable information representation. Therefore, awarding a model for returning cherry-picked relations should not be the actual goal of scene graph generation.

SGG methods are limited by datasets Recent works on scene graph generation methods have mostly explored different strategies to modify network architectures for increased performance. Although these are notable improvements, we argue that scene graph performance is most notably limited by the available scene graph

datasets. This observation aligns with insights gained from other domains in computer vision like object segmentation [60] or depth estimation [139], where large improvements mostly resulted from enhanced training data.

Scene graph generation aims to provide a structured representation of a scene for downstream applications. As such, we argue that the representation should be both correct and exhaustive. However, existing scene graph datasets fail to provide either of those since they focus on salient relations. As a consequence, SGG models falsely learn to ignore actually existing relationships without having an impact on benchmark scores. We therefore argue that blindly improving benchmark scores without addressing the dataset limitations is not useful. In this thesis we will propose new datasets that are designed to provide enhanced representations that provide a more useful structure.

1.2 Contributions

In this thesis, we tackle the outlined challenges and provide methods and datasets that are specifically constructed to fill in the gaps. The contributions of this thesis can be summarized as follows:

Evaluation framework We perform a thorough review of commonly used metrics in scene graph generation. To address the issue of vaguely defined scene graph metrics in the literature, we present rigorous definitions with mathematical equations and pseudocode listings. In addition, we introduce a lightweight and easy to use Python package that supports the discussed metrics. Furthermore, we present a web service that provides a public benchmarking website for scene graph evaluation powered by the introduced Python package. The code and website access can be found here: `https://lorjul.github.io/sgbench`.

Uncovering benchmarking flaws in recent works and addressing them using a new model architecture We analyze the existing evaluation protocols that are used for panoptic scene graph generation (PSGG) and illustrate the inherent flaws. Based on the findings, we introduce a new and improved evaluation protocol that addresses the shortcomings o the current standard. We evaluate existing methods again using the proposed evaluation protocol and discover that recent research trends were based on faulty evaluations. Given these insights, we develop a new neural network architecture for SGG that is easy to train and significantly outperforms prior state-of-the-art methods. The project page including the code can be found here: `https://lorjul.github.io/fair-psgg`.

Improved scene graph evaluation using the Haystack dataset We develop a semi-automatic scene graph annotation pipeline that can be used to efficiently create scene graph datasets with a focus on specific rare predicate classes. The pipeline uses model-guided proposals from predicted relations to identify rare predicates from

a large pool of unlabeled images. Using the proposed pipeline, we construct the Haystack dataset which has favorable properties for fine-grained evaluation of SGG methods. More specifically, the dataset contains explicit negative annotations when a predicate is not the right choice for a certain relation. This enables us to construct a set of new metrics which provide more insight into SGG model performance, especially for rare predicate classes. The project page including the code and dataset can be found here: `https://lorjul.github.io/haystack`.

Parametric relations and proto-relations We propose two new paradigms to represent relations in a scene graph. Parametric relations encode additional information like angles or distances into the scene graph and address the imprecision that are inherent in traditional scene graph ground truth labels. Proto-relations encode potential relations within a scene that can occur if new objects are introduced to the scene. This relation type can enable support for enhanced reasoning and planning.

To create scene graph datasets with these new relation types, we develop a pipeline that extracts comprehensive scene graph ground truth data from arbitrary 3D scenes. We create a new large scale synthetic scene graph dataset based on procedurally generated 3D scenes and demonstrate how existing SGG models can be adapted for this dataset. Additionally, we present a simple graph query framework that can be applied to the extracted graphs, showcasing how future work can make use of the extracted graphs for downstream applications. The project page including the code and dataset can be found here: `https://lorjul.github.io/copasg`.

1.3 List of Publications

The following list shows the papers that I've worked on during my time at the lab. Most parts of this thesis, including figures and tables are taken from these published works. Each chapter includes an explicit reference to the underlying publication if available.

Haystack: A Panoptic Scene Graph Dataset to Evaluate Rare Predicate Classes [79], **Julian Lorenz**, Florian Barthel, Daniel Kienzle, and Rainer Lienhart, *2023 IEEE/CVF International Conference on Computer Vision Workshops (ICCVW)*, 2023.

COVID Detection and Severity Prediction with 3D-ConvNeXt and Custom Pretrainings [53], Daniel Kienzle, **Julian Lorenz**, Robin Schön, Katja Ludwig, and Rainer Lienhart, *Computer Vision – ECCV 2022 Workshops*, 2023.

A Review and Efficient Implementation of Scene Graph Generation Metrics [76], **Julian Lorenz**, Robin Schön, Katja Ludwig, and Rainer Lienhart, *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2024.

A Fair Ranking and New Model for Panoptic Scene Graph Generation [75], **Julian Lorenz**, Alexander Pest, Daniel Kienzle, Katja Ludwig, and Rainer Lienhart, *Computer Vision - ECCV 2024: 18th European Conference, Milan, Italy, September 29 - October 4, 2024, proceedings, part LXI*, 2024 (Oral Paper).

On the Importance of Conditioning for Privacy-Preserving Data Augmentation [80], **Julian Lorenz**, Katja Ludwig, Valentin Haug, and Rainer Lienhart, *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) Workshops*, 2025.

CoPa-SG: Dense Scene Graphs with Parametric and Proto-Relations [77], **Julian Lorenz**, Mrunmai Phatak, Robin Schön, Katja Ludwig, Nico Hörmann, Annemarie Friedrich, and Rainer Lienhart, *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) Workshops*, 2025.

DSFlash: Comprehensive Panoptic Scene Graph Generation in Realtime [78], **Julian Lorenz**, Vladyslav Kovganko, Elias Kohout, Mrunmai Phatak, Daniel Kienzle, and Rainer Lienhart, *arXiv preprint 2603.10538*, 2026.

The following list shows the papers that I've contributed to as a co-author.

Detecting Arbitrary Keypoints on Limbs and Skis with Sparse Partly Correct Segmentation Masks [85], Katja Ludwig, Daniel Kienzle, **Julian Lorenz**, and Rainer Lienhart, *2023 IEEE/CVF Winter Conference on Applications of Computer Vision Workshops (WACVW)*, 2023.

All keypoints you need: detecting arbitrary keypoints on the body of triple, high, and long jump athletes [84], Katja Ludwig, **Julian Lorenz**, Robin Schön, and Rainer Lienhart, *2023 IEEE/CVF International Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), June 17 2023 to June 24 2023, Vancouver, BC, Canada*, 2023.

Towards Learning Monocular 3D Object Localization from 2D Labels Using the Physical Laws of Motion [54], Daniel Kienzle, Katja Ludwig, **Julian Lorenz**, and Rainer Lienhart, *2024 International Conference on 3D Vision (3DV)*, 2024.

The STOIC2021 COVID-19 AI challenge: Applying reusable training methodologies to private data [8], Luuk H. Boulogne, **Julian Lorenz**, Daniel Kienzle, Robin Schön, Katja Ludwig, Rainer Lienhart, Simon Jégou, Guang Li, Cong Chen, Qi Wang, Derik Shi, Mayug Maniparambil, Dominik Müller, Silvan Mertes, Niklas Schröter, Fabio Hellmann, Miriam Elia, Ine Dirks, Matías Nicolás Bossa, Abel Díaz Berenguer, Tanmoy Mukherjee, Jef Vandemeulebroucke, Hichem Sahli,

Nikos Deligiannis, Panagiotis Gonidakis, Ngoc Dung Huynh, Imran Razzak, Reda Bouadjenek, Mario Verdicchio, Pasquale Borrelli, Marco Aiello, James A. Meakin, Alexander Lemm, Christoph Russ, Razvan Ionasec, Nikos Paragios, Bram van Ginneken, and Marie-Pierre Revel, *Medical Image Analysis*, 2024.

Adapting the Segment Anything Model During Usage in Novel Situations [114], Robin Schön, **Julian Lorenz**, Katja Ludwig, and Rainer Lienhart, *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2024.

SkipClick: Combining Quick Responses and Low-Level Features for Interactive Segmentation in Winter Sports Contexts [115], Robin Schön, **Julian Lorenz**, Daniel Kienzle, and Rainer Lienhart, *2025 IEEE/CVF Winter Conference on Applications of Computer Vision Workshops (WACVW)*, 2025.

MMMS: Multi-Modal Multi-Surface Interactive Segmentation [113], Robin Schön, **Julian Lorenz**, Katja Ludwig, Daniel Kienzle, and Rainer Lienhart, *2025 IEEE International Conference on Content-Based Multimedia Indexing (IEEE CBMI)*, 2025.

Leveraging Anthropometric Measurements to Improve Human Mesh Estimation and Ensure Consistent Body Shapes [86], Katja Ludwig, **Julian Lorenz**, Daniel Kienzle, Tuan Bui, and Rainer Lienhart, *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, 2025.

HOIverse: A Synthetic Scene Graph Dataset with Human Object Interactions [97], Mrunmai Vivek Phatak, **Julian Lorenz**, Nico Hörmann, Jörg Hähner, and Rainer Lienhart, *2025 IEEE 8th International Conference on Multimedia Information Processing and Retrieval (MIPR)*, 2025.

Uplifting table tennis: a robust, real-world application for 3D trajectory and spin estimation [55], Daniel Kienzle, Katja Ludwig, **Julian Lorenz**, Shin'ichi Satoh, and Rainer Lienhart, *IEEE/CVF Winter Conference on Applications of Computer Vision (WACV) 2026, 6-10 March 2026, Tucson, AZ, USA*, 2025.

1.4 Thesis Outline

This thesis is structured in seven chapters. Following this introduction chapter, chapter 2 introduces fundamental terminology and other basics like panoptic segmentation and SGG model categories. Additionally, we introduce two commonly used scene graph datasets. Next, chapter 3 dives into the evaluation of SGG mod-

els. As we will show throughout this thesis, a robust evaluation of SGG models is not trivial due to imprecise definitions of metrics and evaluation protocols. Therefore, chapter 3 contains rigorous definitions for a robust and correct evaluation. In chapter 4, we discuss a major issue with prior one-stage SGG models that misinterpreted the evaluation protocol. We address this issue and evaluate these methods again under a fair comparison. Additionally, we present our own DSFormer two-stage SGG model and demonstrate that contrary to recent claims, two-stage methods can indeed outperform one-stage methods if evaluated correctly. In chapter 5, we present our own Haystack dataset that provides a more nuanced evaluation of SGG models. Continuing the goal of improving evaluation and datasets for scene graph generation, we introduce the CoPa-SG dataset in chapter 6. Derived from synthetic 3D scenes, this dataset provides precise and exhaustive ground truth annotations to not only accurately evaluate SGG models but also train SGG models that can be employed in down stream tasks. This thesis concludes with chapter 7 where we summarize the thesis and provide an outlook over the many potential research opportunities that are enabled by the research presented in this paper.

2 Foundations for Scene Graph Generation

In this chapter, we introduce the basics of scene graph generation to better understand the following chapters. We define required terminology, introduce panoptic segmentation and segmentation models, and discuss existing SGG models.

2.1 Terminology

In this section, we define the terminology that is used throughout the thesis. It is mainly based on the following publications:

A Review and Efficient Implementation of Scene Graph Generation Metrics [76], **Julian Lorenz**, Robin Schön, Katja Ludwig, and Rainer Lienhart, *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2024.

Instance An instance refers to a visual object in an image. It is identified by a segmentation mask (or bounding box) and a class label. We define M_{gt} as the set of ground truth instances in an image and M_{out} as the set of predicted instances in an image. We use the term "instances" instead of "objects" to avoid confusion with subjects/objects on relations.

Predicate A predicate is a single label that can be used to describe a relation between a subject instance and an object instance (e.g., "holding", "looking at", "parked on", ...). We define P as the set that contains all possible predicate classes. Note that most relations between instances can be annotated with multiple predicates. For example, in the PSG dataset [137], a person can be "standing on", "on", and "looking at" a road simultaneously.

Relation Triplet We define a relation triplet as a 3-tuple of a subject instance, a predicate label, and an object instance. If it is a ground truth triplet, we define it as $(sbj, predicate, obj) \in M_{gt} \times P \times M_{gt}$. If the triplet was returned by an SGG model, we define it as $(sbj, predicate, obj) \in M_{out} \times P \times M_{out}$. For a triplet t, we use t_{sbj} to refer to the related subject, t_{obj} for the object, and $t_{predicate}$ for the respective predicate category.

A subject and an object can have multiple relations with different predicate classes.

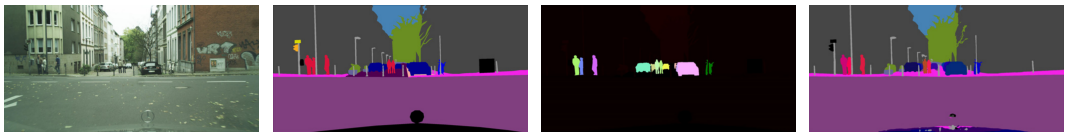
Scene Graph A scene graph is a graph representation that encodes interactions between visual objects in an image. It consists of a set of instances M that are connected by a set of relations, defined as relation triplets $R \subset M \times P \times M$. The scene graph is the pair of both: $\mathbb{G} = (M, R)$, with M being either M_{gt} or M_{out}.

Panoptic Scene Graph While traditional scene graph generation (SGG) typically describes instances using bounding boxes, panoptic scene graph generation (PSGG) [137] uses panoptic segmentation masks instead. These graphs are referred to as "panoptic scene graphs" or simply scene graphs if the context is clear.

2.2 Panoptic Segmentation

As mentioned in the previous section, we have to represent instances in an image using either bounding boxes or segmentation masks. This thesis focuses mostly around segmentation masks.

Image segmentation can be grouped into three major tasks [69], as shown in fig. 2.1: semantic segmentation, instance segmentation, and panoptic segmentation [59].



(a) RGB Image **(b)** Semantic Seg. **(c)** Instance Seg. **(d)** Panoptic Seg.

Figure 2.1: Visual comparison of the three major image segmentation tasks. Adapted from Li and Chen [69].

Semantic segmentation assigns a class to each pixel on an image, illustrated in fig. 2.1b. In instance segmentation, the task is to identify individual, countable instances in an image by assigning separate masks to them. This will exclude areas like roads, grass, or the sky, as shown in fig. 2.1c. The included classes for instance segmentation are also referred to as *thing classes*.

Panoptic segmentation [59] combines semantic segmentation and instance segmentation as one unified task. Here, the goal is to identify individual instances in an image and also classify them. Additionally, panoptic segmentation differentiates between *thing* and *stuff classes* of the segmentation masks. All the *thing classes* from instance segmentation will be used for countable objects such as people, cars, or street signs. The remaining areas in the image are covered by *stuff classes* that were introduced for semantic segmentation. Consequently, multiple segmentation masks with the same *stuff class* are usually merged into one single mask, which usually represent areas of similar texture or material across the image. As such, *stuff classes* cover surfaces of semantically defined non-countable classes of pixels. In practice, *thing classes* and *stuff classes* are treated the same in SGG models and

are just interpreted as segmentation masks with class labels. Both *things* and *stuff* are eligible to be subject or object instances in relation triplets.

2.2.1 Panoptic Segmentation Models

A subcategory of SGG methods are two-stage methods which will be properly introduced in the next section (2.3.1). These methods rely on a panoptic segmentation model to produce segmentation masks which are then processed to return the final scene graph for a given image. As we will show in section 4.3.2, a strong segmentation model is absolutely crucial for the overall performance on scene graph tasks. To retrieve panoptic segmentation masks, we employ pretrained panoptic segmentation models in this thesis, most notably Mask2Former [17], MaskDINO [63], and OneFormer [47]. For a given image, these models return a set of mask-category pairs, where each pair describes one instance in the image. While the mask defines the set of pixels that are associated with the instance, the category of an instance describes what kind of instance it is, e.g., "airplane" or "person".

Mask2Former [17] is the successor of MaskFormer [16]. Similarly to its predecessor, it is a transformer-based neural network designed for universal image segmentation, meaning that Mask2Former's network architecture can be used for semantic segmentation, instance segmentation, and panoptic segmentation. However, it has to be retrained for each task. Mask2Former uses a set of object queries for which a set of segments has to be retrieved. This approach is inspired by DETR [9], which uses object queries for bounding box detection. In Mask2Former, a *transformer decoder* processes the object queries together with a set of feature tensors that represent the respective image.

OneFormer [47] goes one step further. It is an architecture that not only supports the three different segmentation tasks but is also jointly trained on them at the same time. As a consequence, a single model can be used to generate each of the three supported tasks. To differentiate between the current task, OneFormer receives a special task token that tells the model which task should be performed. The joint training strategy helps OneFormer to outperform prior work.

MaskDINO [63] uses a similar approach. It is an adaption of the DINO [142] object detection model[1]. It extends DINO with an additional mask prediction branch. As a consequence, MaskDINO is a unified model that can jointly perform object detection and object segmentation. This unification enables the detection task and segmentation task to mutually enhance each other in a cooperative way. Additionally, MaskDINO benefits from improved training recipes from the DINO object detector.

[1] Not to be confused with the self-supervised pretraining technique of the same name. [10, 95, 120]

2.3 Scene Graph Generation Methods

In scene graph generation, models are typically grouped into two-stage methods and one-stage methods. Two-stage methods first detect instances in an image and then use that information to classify relations between the detected instances. One-stage methods run end-to-end.

2.3.1 Two-Stage Methods

In scene graph generation, two-stage methods split the scene graph prediction into an instance detection stage and a relation classification stage. In the first stage, instances are detected on the input image. This is equivalent to a standard object detection or panoptic segmentation task. Any object detection or panoptic segmentation model is suitable here. In the second stage, the extracted information from the first stage is forwarded to the relation classification stage which predicts the predicate labels of the relations. In most cases, two-stage methods check every possible subject-object combination for potential relations and select the most relevant ones in the end. To achieve good results on SGG benchmarks, the first stage should return segmentation masks and instance classes that are represented in the respective dataset. However, in many cases, the second stage can also process instances that it has not encountered before.

IMP [133] is one of the first SGG models. Starting from a set of instance and relation proposals, IMP uses a recurrent neural network architecture to iteratively refine the proposals. However, as subsequent works have shown, message passing appears to be unnecessary with more capable backbones and datasets. Neural Motifs [140] is an SGG method that is frequently employed in scene graph benchmarks. The authors introduce a fixed frequency bias by counting relation occurrences in the dataset, which improves performance of SGG models. Another prevalent and top-performing model is VCTree [123] which constructs tree structures to represent relations. To generate a scene graph, this tree structure is processed by a tree LSTM. GPS-Net [73] is another SGG method with a highly specialized neural network architecture to improve performance on relation direction and rare predicate classes.

However, as we will show in chapter 4, a much simpler network architecture is capable to outperform these methods.

2.3.2 One-Stage Methods

Contrary to two-stage methods, one-stage methods perform instance detection and scene graph generation in one go. One of the key ideas is that by joining the two tasks together, the model can be trained to primarily detect instances that are involved in relations, thereby increasing scene graph metrics. In section 4.3.2, we will analyze this phenomenon more closely.

Many one-stage architectures are based on or inspired from DETR [9] and HOTR [56]. DETR is a transformer-based architecture for object detection. It receives an

image and a set of object query tokens as input and produces a bounding box prediction for each query token. Kim et al. [56] adapt this approach for human-object interaction and introduce HOTR. In addition to the object query tokens from DETR, HOTR uses additional interaction query tokens that work similarly. PSGTR [137] and PSGFormer [137] adapt the HOTR approach for panoptic scene graph generation. Pair-Net [130] is an extension of PSGFormer and splits the prediction step into pair detection followed by predicate classification. HiLo [146] is another one-stage SGG method that builds on the ideas from IETrans [141]. For each relation, it predicts two predicate distributions. One for low frequency predicates and one for high frequency predicates. The distributions are then merged to one single distribution to tackle the predicate imbalance issue with scene graph datasets.

Although one-stage methods have claimed state-of-the-art performance, we will show in chapter 4 that there are some caveats when evaluating them. One-stage methods skip the instance detection step and are often directly tuned to optimize for scene graph benchmark scores. However, as covered in chapter 4, this approach requires careful post-processing for a fair comparison. If we exclude duplicate masks and relations, two-stage methods may outperform one-stage methods with respect to the scores in the evaluation metrics.

2.4 Scene Graph Datasets

As already outlined in section 1.1, existing scene graph datasets have several critical limitations that we address in this thesis. For the most part, we are using the PSG dataset which is based on Visual Genome.

2.4.1 Visual Genome

One of the first large scene graph datasets used for scene graph generation is Visual Genome [61]. It contains more than 100,000 images where each image is annotated with bounding boxes and relationship annotations between the annotated boxes. Additionally, Visual Genome contains attribute annotations which describe a single bounding box. These attribute annotations are not used in this thesis.

Visual Genome builds on the pioneering work by Johnson et al.[49] and Lu et al.[82] but with a different annotation approach. The authors of Visual Genome observed that human annotators tend to annotate easy and frequently occurring relations if the task is to annotate relations directly from an image as it was done before [49, 82]. To promote relations which are more important for human interpretation of a scene, the authors introduce an intermediate step to focus the annotations on salient relations. First, workers are tasked to draw boxes which contain interesting or relevant content of the image and add an additional description to each box. Next, after a filtering and cleanup step, workers have to annotate all the objects that are mentioned in the descriptions using bounding boxes. Finally, a set of relations is annotated to cover the original descriptions which are then combined into a single

scene graph for the whole image. For a more detailed description of the annotation pipeline, please refer to section 4 of the original paper [61].

Using this approach Visual Genome manages to contain more interesting and relevant relations in the ground truth. However, the original Visual Genome dataset also has some notable downsides. The dataset contains 33,877 different object classes and 40,480 different predicate classes which often overlap in their semantic meaning. These labels are mostly raw labels by annotators with only very slight data post-processing. For example, Visual Genome contains more than 50 different predicate labels that are all synonymous to "A wears B".

To improve this, Xu et al. took Visual Genome and constructed the commonly used VG-150 [133] variant, keeping only the most frequent 50 predicate classes and 150 object classes. Although this variant drastically reduced the number of different predicate classes to the most relevant ones, many predicates are still redundant.

2.4.2 PSG

Yang et al. identified these issues and created the PSG [137] dataset. It is based on the intersection of images from Visual Genome and COCO and contains 48,749 images with panoptic segmentation masks and a total of 56 predicate classes. The authors use the relation annotations from Visual Genome and merge them with the panoptic segmentation masks from COCO. For a given image that is contained in both datasets, Visual Genome instances are matched to their COCO counterparts, totaling to 133 different instance categories. First, the IoU between every instance from the COCO annotation and every instance from the Visual Genome annotation is computed. Next, the following steps are performed until no valid pair of Visual Genome and COCO instances is left:

1. Select the instance pair with the highest remaining IoU.
2. If their categories are similar, the two instances match. They are added to the candidate pool.
3. If their categories are not similar, the two instances are marked as invalid. They are ignored form further processing.
4. Continue at step 1 as long as the highest remaining IoU is nonzero.

To quantify if two categories are similar, a text embedding model [7] is applied on the class names from Visual Genome and COCO. Two categories are considered similar if the cosine similarity between the embeddings is above a certain threshold.

After automatically merging Visual Genome and COCO, the dataset now provides scene graph annotations with panoptic segmentation masks. However, the resulting intermediate dataset is very noisy. Therefore, the authors perform an additional manual verification step where human annotators check the merged dataset for errors.

Instead of using the predicate classes from Visual Genome, a new set of predicate classes is used. The authors aimed to define a set of predicates that are semantically orthogonal. To achieve this, they first consolidated predicates with overlapping or equivalent meanings. For instance, spatial relations such as "parked alongside", "parked behind", and "parked on" are all represented by the single predicate "parked on" in PSG. Similarly, bidirectional predicates like "in front of" and "behind" are merged into a single directional form ("in front of"). This consolidation ensures that the resulting set of predicates is orthogonal, with each predicate representing a distinct and non-overlapping semantic relation. To further improve PSG, the annotators are tasked to use the new predicate classes and annotate additional relations on the images.

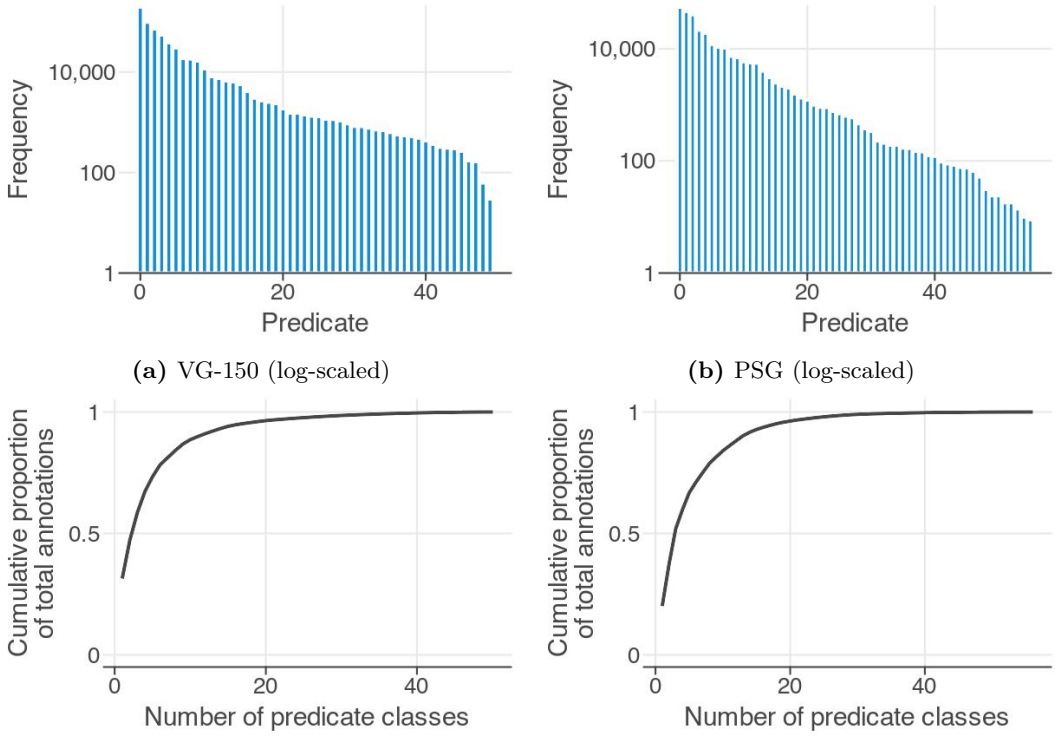
As a much more verified and curated dataset, we choose the PSG dataset as the reference dataset for the majority of this thesis. However, PSG still explicitly focuses on salient relations and many subject-object pairs are unlabeled. Additionally, it also suffers from long-tail distribution of predicates which is discussed in the following section.

2.4.3 Limitations

Implicit negative annotations To learn whether certain relations exist or not in an image, an SGG model must be trained with positive (relation exists) and negative (relation does not exist) ground truth data. Commonly used scene graph datasets contain many different relation annotations per image, and as such contain many positive data samples. However, they do not contain negative annotations (i.e., "there is no relation between this subject and object") and most subject-object pairs remain unlabeled. Since negative annotations are required for training, most SGG methods have to use a fallback solution and assume that missing relation labels can be implicitly interpreted as negative annotations. This assumption works as long as the task is to evaluate how good SGG methods are at retrieving the relations that are most striking to a human annotator. If a comprehensive scene graph annotation is desired, this assumption does not hold anymore. In chapters 4 and 5 we will tackle this limitation and consider datasets that avoid using implicit negative annotations and instead provide negative annotations explicitly.

Long-tail predicate distribution The relation annotations in scene graph datasets follow a long tail distribution, as shown in figs. 2.2a and 2.2b. As illustrated in figs. 2.2c and 2.2d, only a few predicate classes make up most of the total annotations. In fact, an SGG model that can perfectly identify the three most common predicates, will cover more than half of the overall annotations. For example, the three most frequent predicates of the PSG dataset, "on", "beside", and "over", account for 52% of all predicate labels. Furthermore, over a quarter of all predicate classes occur less than 100 times in the dataset.

To address this issue, several approaches have been suggested. First, SGG methods are typically evaluated using *mR@k*, a metric that weighs each predicate class



(a) VG-150 (log-scaled)

(b) PSG (log-scaled)

(c) VG-150

(d) PSG

Figure 2.2: Visualization of the long-tail problem for the VG-150 and PSG scene graph datasets. VG-150 is a curated subset of Visual Genome. Figures (a) and (b) show the distribution of predicate annotations in the respective dataset, ordered from common to rare. Figures (c) and (d) illustrate the increasing proportion of total relation annotations as more predicates are included, starting with the most common and progressing to the rarest. Only a few predicate classes make up the majority of all relation annotations.

the same. However, because of the long-tail distribution, there are only very few samples of rare predicate classes in the test set. Consequently, evaluation is unreliable and volatile for those predicates, influencing the overall metric. The metrics will be covered in more detail in chapter 3. In addition, previous work has come up with various approaches that try to address the issue through techniques such as resampling [24], reweighting [134], or predicate grouping [26]. Another approach is to automatically relabel [66, 141, 146] certain predicates during training from general predicate classes to more specific ones, for instance relabeling "on" to "standing on".

Nevertheless, these methods are ultimately constrained by the same limited test sets which offer very few samples for rare predicate classes. In chapter 5, we present an approach that enables a much more robust evaluation through a specialized evaluation dataset.

3 Evaluation of Scene Graph Generation Methods

This chapter examines the evaluation protocols and metrics commonly used in scene graph generation. We review these metrics in detail and provide precise definitions for each. Our goal is to clarify what each metric measures, as well as its limitations to help readers understand both the strengths and weaknesses of the various approaches. We also develop complementing metrics that address the gaps in existing metrics for a more comprehensive assessment of SGG performance. Then, we introduce the *SGBench* Python package, containing efficient implementations to calculate the defined metrics. Contrary to existing implementations, *SGBench* is framework-agnostic and relies on standard file formats. Finally, we present a benchmarking service for scene graph generation where scene graph generation methods can be conveniently compared against each other.

This chapter is mainly based on the following publication:

A Review and Efficient Implementation of Scene Graph Generation Metrics [76], **Julian Lorenz**, Robin Schön, Katja Ludwig, and Rainer Lienhart, *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2024.

The accompanying code can be found here: `https://lorjul.github.io/sgbench`.

3.1 Introduction

To provide a thorough understanding of scene graph evaluation, we discuss and define the employed metrics in detail in this chapter. This includes the various existing evaluation protocols and the multitude of different metrics that are used to evaluate SGG methods. As covered in chapter 4, imprecisely defined metrics can lead to distorted benchmarks and incorrect conclusion in recent works. In fact, a precise and formal definition of scene graph metrics has long been missing. Since Recall@k ($R@k$), the first metric for scene graphs was introduced by Lu et al. [82], many surveys have covered the topic without rigorously defining the used metrics. Chang et al. [12] briefly examine $R@k$, Mean Recall@k ($mR@k$), and No Graph Constraint Recall@k ($ngR@k$) in a single paragraph, but prioritize on architectural comparisons rather than the employed metrics. Cheng et al. [18] provide a superficial introduction of $R@k$ while only briefly mentioning $mR@k$. Agarwal, Mangal, and Vipul [3] offer descriptions for $R@k$, $mR@k$, and $ngR@k$ without formal definitions.

In contrast, Li et al. [64] provide mathematical definitions for *R@k*, *mR@k*, and *ngR@k*, though with less depth than presented in this thesis. Furthermore, they introduce Zero-Shot Recall@k, which is defined as *R@k* for unseen relation triplets. Given that this metric essentially represents *R@k* on a modified test set, we do not consider it a separate metric in this thesis. Contrary to existing papers, we provide equations and pseudocode to rigorously and unambiguously define *R@k*, *mR@k*, *Pair Recall@k*, and more metrics.

To complement the introduced definitions, we have introduced an efficient implementation, called *SGBench*. Our implementation runs faster than existing implementations and uses a file format that requires less disk space compared to file formats used by prior work. The API is designed to be adaptable, intuitive to use with less required boilerplate code. Additionally, we create a benchmarking website that evaluates SGG models and presents them as a leaderboard. The code and the access to the leaderboard can be found here: `https://lorjul.github.io/sgbench/`.

To summarize, this chapter discusses the following contributions:

1. A thorough **review of commonly used metrics** in scene graph generation with precise definitions.

2. *SGBench*, a lightweight, easy-to-use **Python package** that supports all introduced metrics.

3. A **benchmarking service** where SGG models can be evaluated and compared.

3.2 Protocols

Evaluation of scene graph generation methods is usually classified in three different protocols [133], which are depicted in fig. 3.1. These protocols define what ground truth information is provided to an evaluated model during inference. Failing to recognize these distinctions can result in significant inaccuracies, as we point out in chapter 4.

3.2.1 Scene Graph Detection (SGDet)

The most general protocol is SGDet. In this protocol, only the image itself is provided to the model, without any instance labels or location information. The task is to estimate the locations and class labels of the instances and the predicate labels of the relations in an image. SGDet has also been referred to in the literature as “Scene Graph Generation”. However, we opt to use SGDet to better distinguish between the evaluation protocol itself and the general research area of scene graph generation. SGDet is typically employed to evaluate one-stage methods, which are designed to be end-to-end.

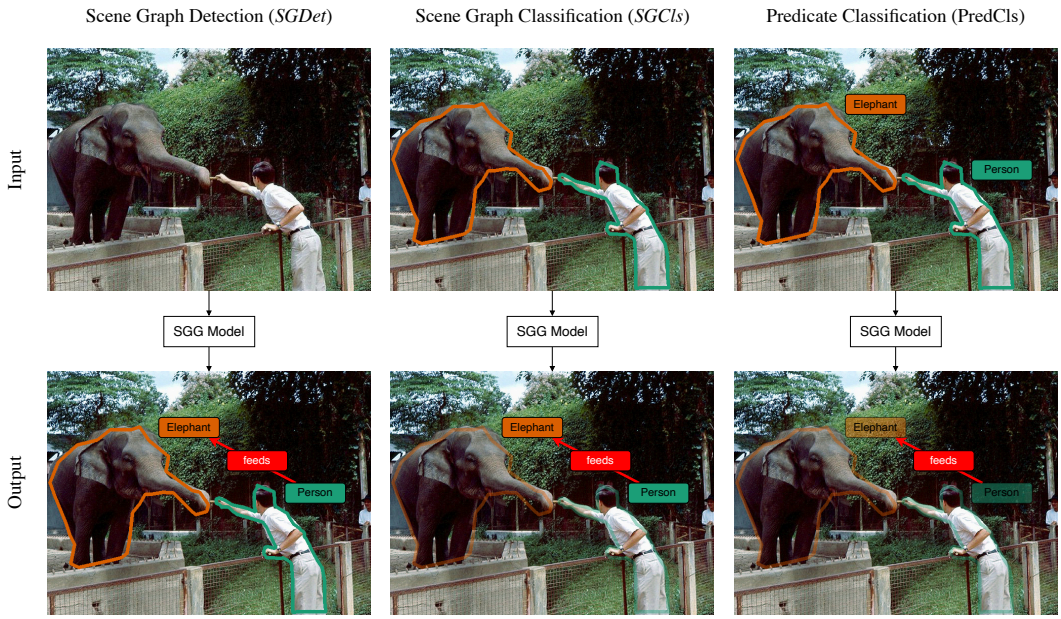


Figure 3.1: Comparison of the commonly used evaluation protocols. SGDet requires the model to predict segmentation masks, instance labels, and relations, while SGCls and PredCls provide some information upfront.

3.2.2 Scene Graph Classification (SGCls)

SGCls is similar to SGDet except that in this protocol, the location information of the instances in an image are provided to the model. However, the class labels of the instances are not provided and have to be predicted, as well as the predicate labels of each relation.

3.2.3 Predicate Classification (PredCls)

PredCls goes one step further and provides instance locations and class labels to the model. The task then only involves classifying the predicate labels of each relation.

The subsequent sections contain metric definitions for the PredCls protocol. In section 3.3, we demonstrate how the metric definitions can be also be applied to SGDet and SGCls.

3.3 Instance Association

To compute scene graph metrics, we compare predicted relations and ground truth relations. The ground truth relations are represented as triplets in $M_{gt} \times P \times M_{gt}$. If the PredCls protocol is used, the model receives ground truth instance class labels and locations as input. Therefore, evaluated models return predicted relations that are also in $M_{gt} \times P \times M_{gt}$. To compute scene graph metrics, it is convenient if the

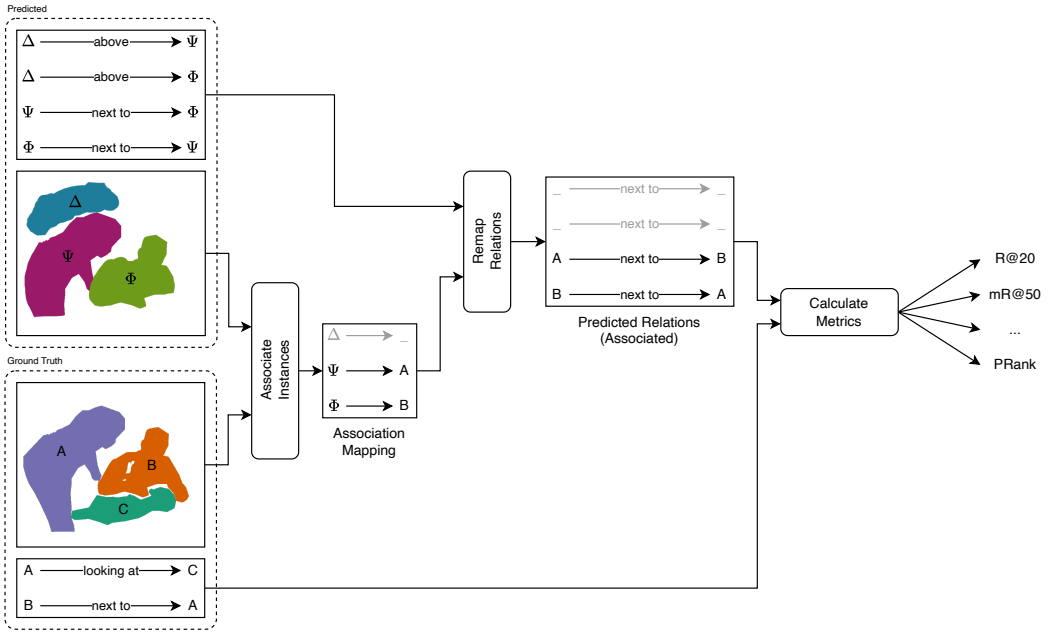


Figure 3.2: Workflow of the instance association step. Since all metrics are defined for the PredCls protocol, this step is necessary to generalize the definitions for SGDet and SGCls. First, the predicted instances (masks or boxes) are associated to ground truth instances by comparing their IoU scores. Next, the obtained association mapping is applied to transform the predicted relations to associated relations. These relations can then be evaluated using the metrics defined in section 3.5. Note that in this process, some predicted masks (e.g., Δ) cannot be associated with any ground truth. Any relations that connect to these masks will count as false positives (e.g., Δ above Ψ).

predicted relations and the ground truth relations are subsets of the same superset. Therefore, we define the metrics in section 3.5 based on the assumption that the ground truth relations and predicted relations are both subsets of $M_{gt} \times P \times M_{gt}$.

However, in the case of the SGDet and SGCls protocols, this is not possible because the predicted instances are in M_{out} and consequently, the predicted triplets are in $M_{out} \times P \times M_{out}$ instead of in $M_{gt} \times P \times M_{gt}$. Therefore, we define a prior instance association procedure to generalize the defined metrics for SGDet and SGCls. By associating M_{out} to M_{gt}, we can derive a remapped relation prediction that is in $M_{gt} \times P \times M_{gt}$ and therefore enables the metrics from section 3.5.

As depicted in fig. 3.2, the instance association process consists of two steps. First, the predicted instances are associated with ground truth instances. Second, the predicted relations are updated to reference the associated ground truth instances.

Algorithm 1 shows how predicted instances are associated with ground truth instances. First, each predicted instance is associated with the ground truth instance that has the same class label and the highest overlap out of all ground truth instances (but at least above a fixed threshold). Typically, a threshold of 0.5 is used.

Algorithm 1 Create Mapping to Associate Instances [76]

Input: Predicted instances M_{out}, ground truth instances M_{gt}, and minimum IoU threshold t
Output: Mapping L that associates instances from M_{out} to instances in M_{gt}
procedure GetMapping(M_{out}, M_{gt}, t)
    Initialize mapping J with $J[x] = null \ \forall x \in M_{gt}$
    for all m in M_{out} **do**
        $M'_{gt} \leftarrow$ subset of M_{gt} that contains only instances with the same class label as m
        $x \leftarrow \arg\max_{g \in M'_{gt}} iou(g, m)$
        if $iou(x, m) > t$ **then**
            if $J[x] = null$ or $iou(x, m) > iou(x, J[x])$ **then**
                $J[x] \leftarrow m$
            end if
        end if
    end for
    $L \leftarrow J^{-1}$ ▷ Use the inverse mapping of J
    return L
end procedure

The overlap between two instances is quantified using intersection over union (IoU). Depending on the type of available ground truth in the respective dataset, IoU is either applied to bounding boxes or segmentation masks. To ensure a one-to-one association, only a one predicted instance can be associated with any ground truth instance. When multiple predicted instances are associated to the same ground truth, the instance which exhibits the highest IoU is selected, while the remaining overlapping instances are discarded and assigned to no ground truth instance.

Following the instance association step, a mapping step is applied to transform the selected predicted relation triplets to ground truth triplets, as detailed in algorithm 2. After this step, some predicted triplets may target invalid masks (if a predicted instance could not be associated). These triplets will count as false positives.

3.4 Converting Scores to Triplets

Current metric implementations are designed to directly process the estimated predicate confidence scores of a neural network instead of a ranked list of relation triplets from a model. This design choice removes the flexibility for a model to independently rank the predicted triplets by their confidence. Some SGG models might have different output structures and therefore rely on different ranking mechanisms. To ensure this flexibility, we define the metrics such that they require a ranked list of

Algorithm 2 Apply Instance Association [76]

Input: Mapping $L: M_{out} \rightarrow M_{gt}$ (algorithm 1), selected triplets X (section 3.4)
Output: Remapped triplets
procedure ApplyAssociation(L, X)
    $X' \leftarrow \emptyset$
    for all t in X **do**
        if $t_{sbj} \in L$ and $t_{obj} \in L$ **then**
            $t' \leftarrow (L[t_{sbj}], t_{predicate}, L[t_{obj}])$
            $X' \leftarrow X' \cup \{t'\}$
        end if
    end for
    return X'
end procedure

predicted triplets as inputs. The definitions assume that the most confident relation comes first in the sorted list.

Converting output scores from a model to a ranked list of relation triplets is left to the model and not part of the metric. Usually, model outputs are converted as follows. Let s be the confidence score associated with predicate p on a relation between a subject sbj and an object obj. This relationship can then be represented as (sbj, p, obj), which can be sorted based on the corresponding confidence score s. *SGBench* contains scripts to perform this conversion automatically.

3.5 Metrics

All discussed scene graph metrics are based on a similar structure as follows (also depicted in algorithm 3):

1. **Triplet subset selection.** A subset of predicted triplets is chosen. The selection depends on the metric being used (e.g., many metrics use a k parameter to limit the number of allowed triplets per image).

2. **Ground truth association.** The selected triplets from step 1, which at this step reference predicted instances, are modified to reference ground truth instances. This involves the association procedure as detailed in section 3.3. Any triplets that reference invalid instances are regarded as false positives and removed from the selection in this step, thereby reducing a model's chance to cover all ground truth relations.

3. **Calculation of image-level scores.** The matched triplets are then compared with the ground truth triplets to calculate an individual score for each image. The details of the calculations vary depending on the metric being used and are explained in the respective sections.

Algorithm 3 General Metric Framework [76]

Input: Output triplets X_i in descending order from an SGG model for each image i in the dataset, ground truth triplets G_i for each image, metric function f at image-level
Output: Metric score for the whole dataset
procedure FRAMEWORK(f, G, X)
    $S \leftarrow \emptyset$
    for all images i in the dataset **do**
        $X' \leftarrow$ relevant subset of X_i
        $L \leftarrow GetMapping()$ ▷ algorithm 1
        $X' \leftarrow ApplyAssociation(L, X')$ ▷ algorithm 2
        $s \leftarrow f(G_i, X')$ ▷ Calculate metric
        $S \leftarrow S \cup \{s\}$
    end for
    return $\frac{1}{|S|} \sum_{t \in S} t$ ▷ Average of S
end procedure

4. **Final score.** Finally, the individual image scores are averaged to determine the overall score for the whole dataset.

Given that steps 2 and 4 remain consistent across all metrics, we define each metric based on its triplet selection procedure and score calculation in sections 3.5.1 to 3.5.5.

3.5.1 Recall@k ($R@k$)

Recall@k is one of the first metrics that was introduced for scene graph generation [82]. It assesses a model's ability to retrieve ground truth triplets using the top k triplet predictions. Calculating $R@k$ only requires counting true positives and false negatives. This is crucial for scene graph generation, because most datasets often lack explicit negative ground truth annotations which are not required for $R@k$.

Triplet Selection

For $R@k$, a model must provide its k most confident triplets, denoted as X_k. As already mentioned before, we leave this step to the design of the evaluated model. A key constraint with $R@k$ is that within X_k, no two triplets are allowed to share the same subject and object. If, for example, a model predicted *(man1, looks at, watermelon2)*, the triplet *(man1, eats, watermelon2)* is prohibited for this metric.

Score Definition

To compute an image-level score, the recall between the predicted triplets X_k and ground truth triplets G is calculated:

Algorithm 4 Mean Recall for a Single Image [76]

Input: Set of all predicate classes P, ground truth triplets G, matched triplets X_k (algorithm 2)
Output: Mean Recall@k
procedure MEANRECALL(P, G, X_k)
    $P' \leftarrow$ subset of P that contains only predicates that exist in G
    for all predicate classes p in P **do**
        $G^{(p)} \leftarrow \{\, t \in G \mid t_{predicate} = p \,\}$
        $X^{(p)} \leftarrow \{\, t \in X_k \mid t_{predicate} = p \,\}$
    end for
    return $\frac{1}{|P'|} \sum_{p \in P'} \frac{|G^{(p)} \cap X_k^{(p)}|}{|G^{(p)}|}$
end procedure

$$R@k = \frac{|G \cap X_k|}{|G|} \tag{3.1}$$

3.5.2 Mean Recall@k ($mR@k$)

Mean Recall@k [15, 123] tackles the issue of predicate imbalance [14, 65, 67, 70, 141, 146] in typical scene graph datasets. It achieves this by calculating a $R@k$ score for each predicate individually and then taking the mean with equal weighting. This approach ensures that rare predicates influence the final score with the same impact as common predicates. The whole process is detailed in algorithm 4.

Triplet Selection

$mR@k$ shares its triplet selection process with $R@k$, and is exactly the same as described in section 3.5.1. Per image, the top k predicted triplets are selected, without allowing duplicate subject-object combinations.

Score Definition

To compute the score for each predicate individually, the set of ground truth triplets G and the set of selected triplets X_k are both first divided into subsets based on the associated predicates. Let P' be the subset of predicate classes that can be found in the ground truth triplets G. For each predicate $p \in P'$, the ground truth triplets G are split into individual sets $G^{(p)}$. Similarly, X_k is split into $X_k^{(p)}$. It's important to note that typically $|X_k^{(p)}| < k$ for each $p \in P'$, since the k elements of X_k are distributed across the individual subsets.

Finally, the image-level score is calculated as:

$$mR@k = \frac{1}{|P'|} \sum_{p \in P'} \frac{|G^{(p)} \cap X_k^{(p)}|}{|G^{(p)}|} \tag{3.2}$$

3.5.3 Pair Recall@k ($PR@k$)

Pair Recall@k [130] is an alternative to $R@k$ that focuses solely on a model's ability to retrieve ground truth relations, disregarding the predicted predicate for the relations.

Triplet Selection

$PR@k$ employs the same triplet selection process as $R@k$, discussed in section 3.5.1.

Score Definition

$PR@k$ is calculated very similarly to $R@k$ except that the predicate labels are disregarded. Let G' be the set of subject-object pairs that is returned by removing the predicate labels from the ground truth triplets G, as depicted in eq. (3.3). The same process is applied to the selected triplets X_k, resulting in X'_k, shown in eq. (3.4). Finally, $PR@k$ is calculated similarly to $R@k$, as defined in eq. (3.5).

$$G' = \{(s, o) \mid (s, p, o) \in G\} \tag{3.3}$$

$$X'_k = \{(s, o) \mid (s, p, o) \in X_k\} \tag{3.4}$$

$$PR@k = \frac{|G' \cap X'_k|}{|G'|} \tag{3.5}$$

3.5.4 No Graph Constraint Recall@k ($ngR@k$)

No Graph Constraint Recall@k ($ngR@k$) [93, 140] relaxes the usual constraint of having only one triplet per subject-object pair in the set of predicted triplets. Given the predicates are different, multiple predicted triplets are allowed per pair. This enables $ngR@k$ to evaluate a model's ability to provide alternative or "second guess" relations.

The difference between the $R@k$ and $ngR@k$ metrics has been the source of misinterpretations that we will further discuss in chapter 4. The key problem is that $R@k$ and $ngR@k$ only differ in which predicted relation triplets are allowed in the selection. In existing metric implementations, it is assumed that the model selects the correct triplets for the respective metric. As we show in chapter 4, this is a risky assumption, since researchers often expect this functionality to be part of the metric implementation. With *SGBench*, we make sure that the triplet selection process is verified and correctly applied in order to prevent such issues.

Triplet Selection

Compared to $R@k$, $ngR@k$ still permits k triplets, but the triplets are allowed to share the same subject-object combination, provided their predicates differ. For example, *(man1, looks at, watermelon2)* and *(man1, eats, watermelon2)* are allowed to be selected together.

Score Definition

Given the described triplet selection process, *ngR@k* score is computed like *R@k* (section 3.5.1) but using a different set of selected triplets X_k.

3.5.5 Mean No Graph Constraint Recall@k ($mNgR@k$)

mNgR@k is a variation of *ngR@k* that assigns equal weight to each predicate class, mirroring the approach of *mR@k*. The score is computed similarly to *mR@k*, but with the triplet selection process from *ngR@k*.

3.5.6 Choice of k

Typically, fixed absolute numbers are chosen for k in the literature, with common choices including 20, 50, and 100. Consequently, the same number of triplets is selected on every image.

It is worth noting that the number of ground truth triplets varies a lot between different images. Having fixed number of predicted triplets can therefore distort individual images scores. For example, an image that contains only three ground truth relation triplets is much easier to cover with 20 relation predictions than an image with 90 ground truth relation triplets.

Therefore, we also include relative values for k in our analysis that depend on the number of ground truth relations in an image. In the following, we use a multiplication symbol to denote relative k, e.g., *R@×10*. If there are for example 13 annotated relations in an image, *R@×10* allows $13 \cdot 10 = 130$ predicted triplets to be selected.

3.5.7 Instance Recall ($InstR$)

InstR is technically not a scene graph metric because it only quantifies the quality of the instance matching process. It measures the proportion of instances that are successfully retrieved by the SGG model. More precisely, *InstR* calculates how many predicted instances M_{out} can be matched to the set of ground truth instances M_{gt}. To this end, the mapping L described in section 3.3 is used.

$$InstR = \frac{|\{m \in M_{out} \mid L[m] \in M_{gt}\}|}{|M_{gt}|} \tag{3.6}$$

3.5.8 Metrics With $k = \infty$

InstR provides a convenient measure to quantify how well the instances in an image are predicted by an SGG model. Another valuable metric is to compute the standard scene graph metrics like *R@k* or *mR@k* with $k = \infty$. The idea behind this metric is that we assume a hypothetical SGG model that is perfect at classifying predicates given a set of instances. Therefore, the final metric score can only decline if some ground truth instances are not detected. The pseudocode for this calculation is shown in algorithm 5.

Algorithm 5 Recall@∞ for a Single Image [76]

Input: ground truth triplets G, mapping $L: M_{out} \rightarrow M_{gt}$ (algorithm 1)
Output: Recall@∞
procedure RECALL@$\infty(G, L)$
    $X' \leftarrow \emptyset$
    $J \leftarrow L^{-1}$ ▷ inverse mapping of L
    for all ground truth triplets t in G **do**
        if $J[t_{sbj}] \neq null$ and $J[t_{obj}] \neq null$ **then**
            $X' \leftarrow X' \cup \{t\}$
        end if
    end for
    return $\frac{|G \cap X'|}{|G|}$
end procedure

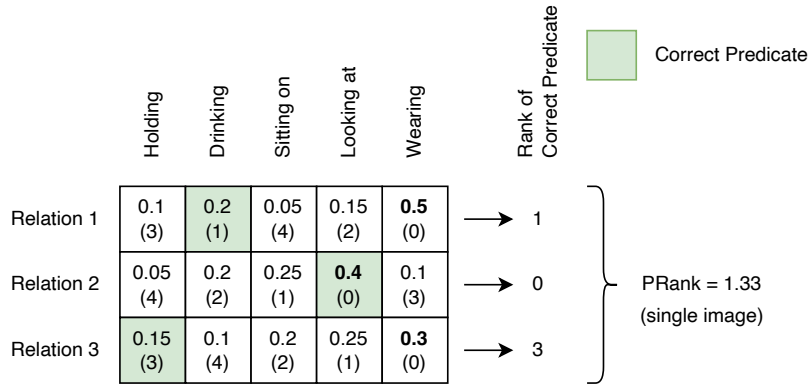


Figure 3.3: Schematic of how *PRank* is calculated for an individual image.

3.5.9 Predicate Rank (*PRank*)

PRank is a complementary metric to *PR@k*, introduced in section 3.5.3. While *PR@k* assesses an SGG model's ability to identify relevant subject-object pairs, disregarding the assigned predicate, *PRank* evaluates how well a model selects the correct predicate class, given a correctly identified subject-object pair.

Figure 3.3 illustrates the principle behind *PRank*. For its final prediction, an SGG model assigns confidence scores to each possible predicate for a given subject-object pair. This allows for sorting the predicates within a pair and retrieving the rank of the ground truth predicate for each pair. *PRank* returns the average rank of the ground truth predicates, with a score of 0 representing the best possible performance.

Pseudocode for *PRank* is presented in algorithm 6. To calculate *PRank*, all successfully matched triplets are used. Initially, all predicted triplets for each subject-object pair are ranked by model confidence and get a per-pair rank assigned. Then, for each ground truth triplet, the predicted rank from the model can be selected.

Algorithm 6 Predicate Rank [76]

Input: Set of all predicate classes P, ground truth triplets G, ordered sequence of all matched triplets X
Output: Predicate Rank
procedure PREDICATERANK($P, G, X, rank$)
  for all predicates p in P **do**
    $G^{(p)} \leftarrow \{t \in G \mid t_{predicate} = p\}$
  end for
  Initialize Φ with $\Phi[t] = null \ \forall t \in X$
  Initialize C with $C[t_{sbj}, t_{obj}] = 0 \ \forall t \in X$
  for all predicted triplets t in X **do** ▷ X is sorted
    $\Phi[t] \leftarrow C[t_{sbj}, t_{obj}]$
    increment $C[t_{sbj}, t_{obj}]$ by 1
  end for
  $R \leftarrow \emptyset$
  for all predicates p in P **do**
    $R^{(p)} \leftarrow \emptyset$
    for all triplets t in $G^{(p)}$ **do**
      if $\Phi[t] \neq null$ **then**
        $R^{(p)} \leftarrow R^{(p)} \cup \{\Phi[t]\}$
      end if
    end for
    $R \leftarrow R \cup \left\{ \frac{1}{R^{(p)}} \sum_{r \in R^{(p)}} r \right\}$ ▷ Add average of $R^{(p)}$
  end for
  return $\frac{1}{|R|} \sum_{r \in R} r$ ▷ Return average of R
end procedure

Averaging the predicted ranks returns the value for *PRank*. In case of a triplet mismatch, the correct predicate cannot be matched, and the triplet is omitted in the calculation.

3.6 Python Package Implementation (SGBench)

We provide one unified metrics implementation for the previously introduced metrics in the *SGBench* Python package. It is built to be used as a lightweight, standalone library. *SGBench* can be easily installed via pip: `https://pypi.org/project/SGBench`.

To achieve this, we limit the number of external dependencies to the following packages:

1. *NumPy* [39]: efficient calculations

2. *Pillow* [88]: loading ground truth segmentation masks from PNG images
3. *tifffile* [34]: loading TIFF files
4. *imagecodecs* [33]: LZMA and Deflate compression methods

Consequently, *SGBench* has no dependencies on machine learning frameworks, which ensures an easy integration of *SGBench* into a wide variety of existing machine learning code bases. Since there are only few dependencies, conflicts are less likely to occur and are easier to upgrade without issues.

Moreover, when creating *SGBench*, we focused on ensuring that the package code is easy to read and modify to individual needs. The implementation achieves this with a streamlined approach, requiring less boilerplate code by the user and fewer lines of code in the *SGBench* implementation. Compared to the metrics implementation from PSG [137], which contains more than 1500 lines of code, *SGBench* contains less than 700 lines, even though supplementary metrics are supported.

3.6.1 File Format

To improve comparability among different SGG models, which often employ custom file formats for model outputs, we develop a unified file format. This file format is designed to be independent of any specific implementation and aims to facilitate interoperability between SGG methods. Furthermore, we include a set of utilities to simplify the conversion of output files to the new format.

3.6.1.1 Triplets File

To ensure broad compatibility, relation triplets are stored as JSON [28]. JSON's human-readable design allows an easy inspection with a text editor, while it also enjoys widespread adaption across different tools and programming languages. In contrast, many scene graph implementations use the Pickle [100] file format, which is not human-readable. Additionally, it is Python-specific and depends on a specific version of Python. If custom data structures are serialized in a Pickle file (as it is common in scene graph implementations), the file can only be deserialized if the reader possesses the required class definitions. This can be cumbersome in practice, because the data structures are often not documented, and users have to fall back to import all dependencies from the code base that wrote the files. Furthermore, the Pickle format has security vulnerabilities that even enable arbitrary code execution [21, 98, 99]. Contrary, JSON itself is secure against such attacks and is completely self-contained without relying on external class definitions.

Listing 1 provides an example of a triplet output file. The `"version"` item is included to facilitate version upgrades with possible future file formats. It is currently fixed at a value of 1. For every image that was processed by the respective SGG model, a new item is added to the `"images"` list, containing an `"id"` that corresponds to the image ID within the ground truth annotation file. This ID is later used to

correctly assign the image to the ground truth counterpart. The `"seg_filename"` property specifies the relative path to the corresponding TIFF file with the predicted segmentation masks. The `"instances"` list details the instance predictions from the model, where each entry aligns with a layer in the TIFF file. Within each instance entry, `"bbox"` represents the respective bounding box in *xyxy* format (a 4-tuple of the inclusive left, top, right, and bottom coordinate of the box). This field is used if segmentation masks are either unavailable or deliberately skipped in evaluation. The `"category"` field indicates the 0-based instance class label index. Finally, each image entry also contains a list of *subject-object-predicate* triplets, stored in the `"triplets"` list. There are two important details about this list. First, the triplets are stored as *subject-object-predicate* instead of *subject-predicate-object*. The idea behind this decision is that similar data should be stored next to each other. Second, the triplets in this list are sorted by decreasing model confidence. The presented file format does not provide means to store actual confidence scores by design, which is justified in sections 3.4 and 3.5. The *subject* and *object* values are 0-based indices referencing items of the instances list. The *predicate* value is a 0-based index that denotes the predicate class label. The index references the label list of the original ground truth file. The number of items in the triplets list is not limited and *SGBench* automatically selects the appropriate triplets subset based on the order of triplets for each metric to evaluate.

3.6.1.2 Segmentation Masks

Panoptic scene graph generation requires additional segmentation masks for evaluation. For the presented file format, predicted segmentation masks are stored as TIFF [2] images. Unlike PNG files, TIFF files can accommodate an arbitrary number of layers, enabling overlapping segmentation masks, which some SGG methods [130, 137, 146] produce. Furthermore, TIFF files enjoy widespread support and result in significantly reduced file size compared to NumPy arrays for the same tasks. To this end, we employ Deflate [25] compression, which can be effectively paired with TIFF files. Alternatively, LZMA [96] compression is another option that is slower but has a higher compression rate.

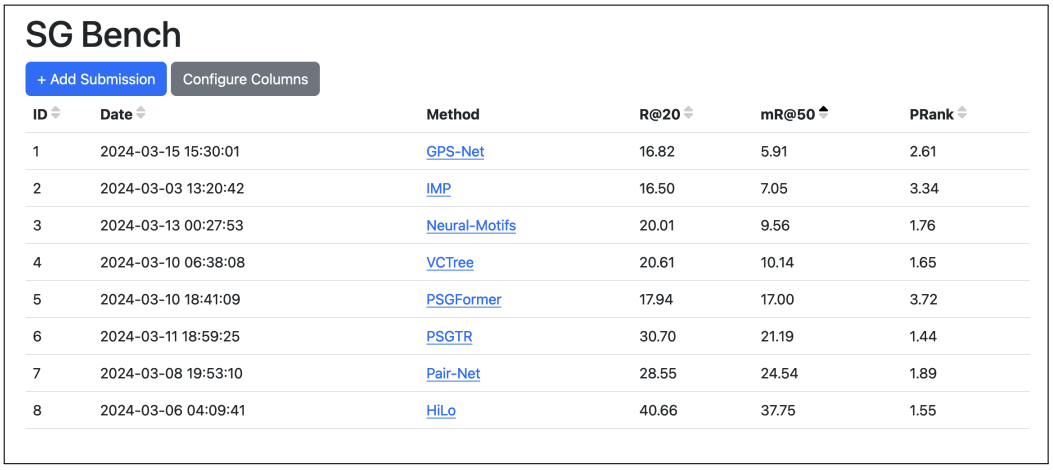
3.6.2 Comparison to the OpenPSG Implementation

In this section, we compare *SGBench* against the commonly used implementation from the PSG dataset [137], referred to as OpenPSG in the following.

Ease-of-use OpenPSG's code for evaluation and metrics is tightly integrated with the rest of the code base, making it challenging to use it in a separate environment as a standalone tool. Contrary, *SGBench* depends on only four external packages, which eases compatibility issues with existing environments. Furthermore, *SGBench* can be conveniently installed using pip.

Listing 1 Example triplet output file [76]

```
{
  "version": 1,
  "images": [
    {
      "id": 123, // image id from the scene graph dataset
      "seg_filename": "seg_file.tiff",
      "instances": [
        {
          "bbox": [1, 22, 333, 44.4],
          "category": 2
        },
        // more instances ...
      ],
      "triplets": [
        [0, 3, 34],
        [2, 0, 13],
        // more triplets, ordered
        // by descending confidence ...
      ]
    },
    // more images ...
  ]
}
```



ID	Date	Method	R@20	mR@50	PRank
1	2024-03-15 15:30:01	GPS-Net	16.82	5.91	2.61
2	2024-03-03 13:20:42	IMP	16.50	7.05	3.34
3	2024-03-13 00:27:53	Neural-Motifs	20.01	9.56	1.76
4	2024-03-10 06:38:08	VCTree	20.61	10.14	1.65
5	2024-03-10 18:41:09	PSGFormer	17.94	17.00	3.72
6	2024-03-11 18:59:25	PSGTR	30.70	21.19	1.44
7	2024-03-08 19:53:10	Pair-Net	28.55	24.54	1.89
8	2024-03-06 04:09:41	HiLo	40.66	37.75	1.55

Figure 3.4: Screenshot of the benchmarking service interface.

Speedup *SGBench* offers significant advantages in terms of performance, supporting multiple CPU cores to accelerate the evaluation. While OpenPSG requires approximately 66 seconds to process output from HiLo, *SGBench* completes the same task in just 20 seconds. Notably, *SGBench* offers improved speedup but also calculates additional metrics which OpenPSG does not support.

Storage efficiency The `-submit` flag from OpenPSG enables a more compact output format. However, it lacks support for overlapping masks which PSGFormer, PSGTR, HiLo, and Pair-Net require. Consequently, the Pickle output format is the only option when using OpenPSG. In our experiments, the Pickle format from OpenPSG consumes 117 GB of disk space when used with Pair-Net. In contrast, *SGBench* offers a significantly smaller footprint of just 761 MB, yet preserves all the data required for evaluation.

3.7 Benchmarking Service

To facilitate convenient comparison between SGG methods, we present a benchmarking website powered by *SGBench*. The website provides an easy-to-use way to evaluate and compare different SGG methods that use the file format, introduced in section 3.6.1. The server-side code and access to the website can be accessed here: `https://lorjul.github.io/sgbench`.

Users intending to upload generated scene graphs to the server must first convert their model output to the specifications detailed in section 3.6.1. This format encompasses multiple files, specifically one JSON triplet file and multiple segmentation masks, stored as TIFF files. To submit, all of these files must be combined into one single ZIP archive, which can then be uploaded to the benchmarking website.

Following submission, the server schedules the evaluation process and subsequently add the results to a publicly accessible leaderboard, as illustrated in fig. 3.4. Users have the option to include hyperlinks with their submission to reference a website entailing more information about the related research. To begin, we have populated the leaderboard with results from recent PSGG methods and added links to the associated sources.

3.8 Conclusion

In this chapter, we addressed a notable gap in the scene graph literature by presenting a comprehensive review of scene graph metrics. The covered metrics include the commonly used ones like *R@k*, *mR@k*, and *ngR@k*, but also newly introduced metrics like *PRank* and *InstR*.

Moreover, we have developed *SGBench*, a Python package to calculate scene graph metrics, designed to be lightweight, efficient and developer-friendly.

4 Benchmarking Flaws and DSFormer Architecture

In section 3.2, we have introduced the different evaluation protocols that are commonly used in scene graph generation. In this chapter, we now analyze how some existing work on panoptic scene graph generation (PSGG) suffers from a flawed evaluation process that is caused by not adhering to those protocols. This evaluation flaw skews the reported scene graph metrics and produces a distorted view on recent state-of-the-art models. In this chapter, we evaluate these models again with a corrected evaluation protocol. In fact, this update reveals significant score adjustments: while two-stage methods see score increases of up to 7.4 *mR@50*, one-stage methods experience score decreases of up to 19.3 *mR@50*. This discrepancy stresses the critical need for accurate evaluation procedures for PSGG. Contrary to recent findings, our analysis demonstrates that two-stage methods remain highly competitive with one-stage methods. Building upon this understanding, we introduce the Decoupled SceneFormer (DSFormer), a two-stage model that achieves substantial improvements compared to prior SGG models, surpassing them by +11 *mR@50* and +10 *mNgR@50*. A key design element of DSFormer is the direct encoding of subject and object masks into the feature space.

This chapter is mainly based on the following publication:

A Fair Ranking and New Model for Panoptic Scene Graph Generation [75], **Julian Lorenz**, Alexander Pest, Daniel Kienzle, Katja Ludwig, and Rainer Lienhart, *Computer Vision - ECCV 2024: 18th European Conference, Milan, Italy, September 29 - October 4, 2024, proceedings, part LXI*, 2024 (Oral Paper).

4.1 Introduction

Previously, PSGG models have been evaluated using the protocol introduced by Yang et al. [137] which we refer to as the *Multiple Masks Per Object Evaluation Protocol* (*MultiMPO*) in this section. This protocol contains two significant flaws that can substantially skew the reported scores, as illustrated in fig. 4.1.

1. The *MultiMPO* allows generated scene graph masks to overlap which renders a one-to-one correspondence between graph nodes and real-world instances impossible.

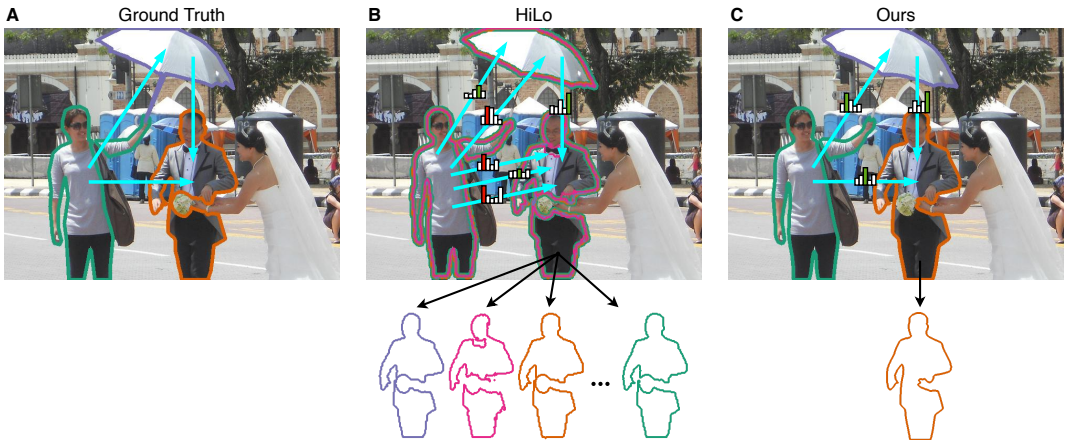


Figure 4.1: A schematic model output comparison between one-stage methods (such as HiLo, illustrated in Fig. B) and our proposed two-stage method (Fig. C). One-stage methods frequently generate multiple masks for individual real-world instances, as shown by the colored masks in Fig. B. This results in a predicate score distribution for each mask pairing, but creates multiple distributions when the pairings share the same ground truth subject and object. Curent implementations fail to aggregate these multiple masks or relations, which can be exploited to artifically inflate *mR@k* scores. Our new SGG methods avoids this issue by design.

2. It permits models to output multiple predicate distributions for the same ground truth subject-object pair. This allows a model to artificially increase its chance of success by repeatedly predicting the same subject-object relation with different predicates, which contradicts the underlying metric definitions.

In this chapter, we demonstrate how to address these issues and propose an updated, more rigorously defined evaluation protocol which we call *Single Mask Per Object Evaluation Protocol* (*SingleMPO*).

Compared to the previous *MultiMPO*, existing one-stage PSGG models now achieve considerably lower *mR@k* scores than previously reported, with a decrease of up to 19.3 *mR@k*. On the other hand, existing two-stage methods demonstrate robustness to the evaluation protocol and can even be improved when combined with a state-of-the-art segmentation model.

Recent trends in scene graph generation have favored one-stage methods, which infer both the graph and masks in a single pass [130, 137, 146]. However, we demonstrate that by leveraging recent advancements in segmentation models, two-stage methods are now able to surpass their one-stage counterparts. Based on these findings, we create the *Decoupled SceneFormer* (DSFormer), a model that is specifically designed to process the outputs of a segmentation model and only focus on inferring the scene graph. This focused design allows us to employ a simpler network architecture, which is straightforward to train and modify. It also significantly outperforms

other networks across various metrics like *mR@20*, *mR@50*, *mNgR@50*, establishing a new state-of-the-art result in PSGG.

We provide the code that was used for training and inference, as well as evaluation scripts for other PSGG using *SingleMPO* models here: `https://lorjul.github.io/fair-psgg`.

Our contributions can be summarized as:

1. We analyze the existing evaluation protocol used for PSGG and illustrate the **inherent flaws**.
2. Based on the findings, we introduce a new and **more accurate protocol** (*SingleMPO*), designed to address the shortcomings of the current standard.
3. Using *SingleMPO*, we provide a comprehensive and **fair re-evaluation** of existing methods.
4. Following on the insight that two-stage methods are in fact competitive, we develop a **new two-stage architecture** (DSFormer) which is straightforward to train and significantly outperforms current state-of-the-art methods, achieving an increase of +11 points on *mR@50*.

4.2 Methods

As usual, we use the scene graph definitions introduced in section 2.1. For this chapter in particular, it is worth remembering that SGG models usually predict a predicate class distribution for every relation and select the predicate with the highest confidence when calculating metrics.

In this section, we outline two essential requirements for a correct evaluation of panoptic scene graphs and analyze how recent models fail to meet them.

4.2.1 Requirements for a Fair Evaluation

Unique Instances A robust SGG model should produce a connected graph of distinct instances. As illustrated in fig. 4.2B, a connected graph is not achievable when multiple instance predictions are being made for the same real-world instance. Therefore, instance predictions are required to be unique; no multiple different instance predictions for the same ground truth instance is allowed.

This is not guaranteed using the *MultiMPO* protocol. It allows multiple masks to be considered correct, even if they are nearly identical, as long as each mask has an IoU greater than 50% with the ground truth.

In contrast, *SingleMPO* enforces the constraint of a single mask per ground truth instance. To aggregate multiple output masks from an SGG model, we employ a process that is similar to non-maximum suppression:

1. Given a set of output masks and associated confidence scores from an SGG model, we build a candidate pool.

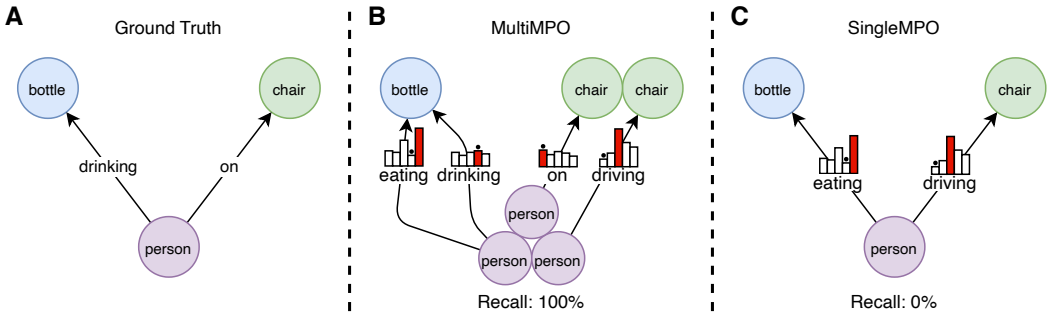


Figure 4.2: Comparison of *MultiMPO* versus *SingleMPO*. (**A**) In the ground truth, each instance has one single mask assigned. (**B**) In this scenario, multiple masks are generated for the same instance (three for "person" and two for "chair"). Consequently, all of the ground truth instances are covered and result in a relation recall of 100% using *MultiMPO*. However, the model in this example is much more confident to predict *person-eating-bottle* than *person-drinking-bottle* and *person-driving-chair* instead of *person-on-chair*, which has no effect at all on the *MultiMPO* recall score. (**C**) By making sure that only a single mask per instance and a single predicted predicate distribution per subject-object pair is used, a recall of 0% is correctly computed.

2. Draw the most confident mask from the candidate pool and add it to the output list.
3. Discard all remaining masks from the pool that have an overlap of 0.5 or greater with the picked mask.
4. Continue at step 2 until the candidate pool is empty and all masks have been processed.

After the remapping process, some relations will share the same subject-object combination, because the associated instances have been assigned to the same ground truth instance. This will be handled in the next step.

Unique Relations Figure 4.2B illustrates a hypothetical example where a model outputs two predicate distributions for each subject-object pair. Although the model has more confidence that the person is *eating* the bottle, instead of *drinking* from it, it achieves a perfect recall score with the *MultiMPO* protocol by using both relation predictions during evaluation. However, using both predictions is not allowed when evaluating using *R@k* or *mR@k*. Nevertheless, recent SGG models ignore this restriction and frequently report this score as *mR@k*, giving them an unfair edge over models that adhere to the single relation constraint.

With the *SingleMPO* protocol, all methods are evaluated fairly, and the recall for the example in fig. 4.2 is correctly calculated as 0%. If multiple predicates are intended during evaluation, the *mNgR@k* metric should be employed instead. It is worth noting that even for *mNgR@k*, a predicate is only allowed to be predicted

once per subject-object pair. Returning the same predicate multiple times for a pair would again distort the calculated metric.

The *SingleMPO* protocol ensures correct triplets by merging the estimated predicate distributions. SGG predictions for a relation are usually represented by a subject, object, and a distribution of predicate confidence scores. Most SGG methods also contain a *no-relation* output that is set to a high value if none of the predicates applies. To merge duplicate relation predictions for a specific subject-object pair, the *SingleMPO* protocol selects the highest confidence score per predicate. Through empirical study, we have observed that instead of choosing the associated predicted *no-relation* score, a better way is to average all *no-relation* scores per subject-object pair. This combination ensures that within a subject-object pair, SingleMPO still follows the models intended output predicate while also aggregating the full model output instead of just discarding the remaining predictions. Let P be the set of all predicate classes. For a given subject-object pair, we define a predicted predicate distribution as $\psi \in \mathbb{R}^{|P|+1}$. ψ has $|P|+1$ coefficients to also accommodate the *no-relation* score which is not part of P. Given m different predicate distributions $\psi^{(1)}, ..., \psi^{(m)} \in \mathbb{R}^{|P|+1}$ for the same subject-object pair, we construct the merged distribution $\overline{\psi} \in \mathbb{R}^{|P|+1}$ using:

$$\overline{\psi}_{norel} = \frac{1}{m} \sum_{i=1}^{m} \psi_{norel}^{(i)} \tag{4.1}$$

$$\overline{\psi}_{p} = \max_{i \in [1,m]} \psi_{p}^{(i)}, \ \forall p \in P \tag{4.2}$$

$$\tag{4.3}$$

Note that for evaluation, we do not require a probability distribution for each subject-object pair. Since we don't want to skew the estimated individual confidence scores from the SGG model, we don't normalize the merged distributions to 1.

Summary These two requirements eliminate ambiguity when mapping the instance and relations of a generated scene graph to the real world. Furthermore, they prevent SGG models from gaining an unfair advantage by returning multiple predicate distributions for each subject-object pair. Unlike the *MultiMPO* protocol, the *SingleMPO* protocol guarantees that these requirements are consistently met.

4.2.2 Model Overview

Leveraging the significant advancements in panoptic segmentation methods [17, 47, 63], we opt for a completely decoupled two-stage approach. The first stage uses an established segmentation model to produce segmentation masks and class labels for a given image. These masks and labels are then fed as additional inputs to our novel Decoupled SceneFormer (DSFormer) architecture. Importantly, only the DSFormer requires training; a fixed pretrained segmentation model, such as MaskDINO, is sufficient.

This approach offers four key advantages:

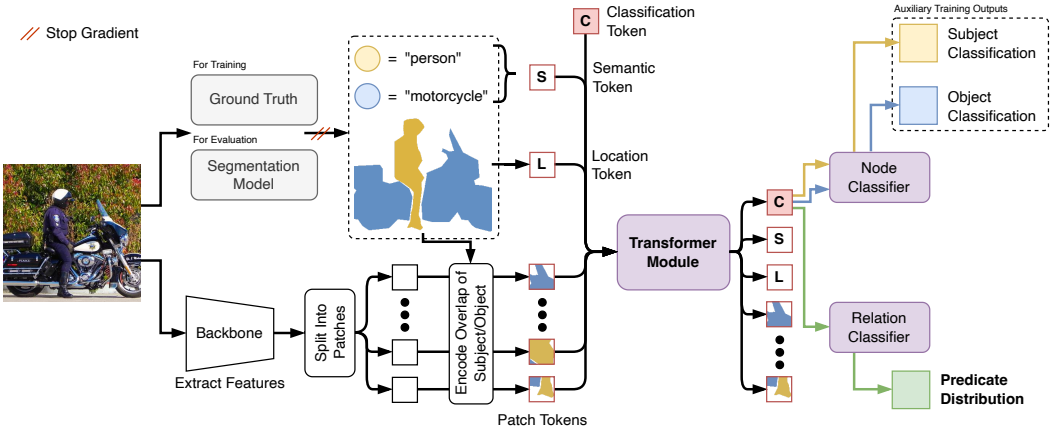


Figure 4.3: Architecture overview of DSFormer. In a forward pass, the model requires an image with corresponding segmentation masks for subject and object, as well as their classes. During training, the masks are sourced from the ground truth. For evaluation, a capable segmentation model provides the masks and class labels. DSFormer's output consists of a relation prediction and an auxiliary subject/object class prediction that are used exclusively for training. Figure 4.4 illustrates how the various tokens for the transformer module are retrieved.

1. With segmentation masks already generated and available, a smaller model can be employed to build the scene graph, reducing computational cost during training and lowering requirements to hardware.

2. Two-stage methods directly benefit from state-of-the-art foundation models for image segmentation without needing to integrate them into the training pipeline. These models are trained on datasets which are significantly larger [60, 72, 117, 131] than those available for scene graph training and will naturally produce superior masks compared to one-stage SGG models.

3. Changing the segmentation model requires minimal effort and eliminates the need for retraining. For a fair comparison of new SGG methods against existing two-stage approaches, it is crucial to replace the first stage model with the same segmentation model that new SGG models use.

4. Two-stage methods can be prompted with specifically selected subject-object pairs during training which enables greater control over sampling strategy and loss weighting during training.

Figure 4.3 illustrates the architecture of DSFormer. While the model receives ground truth segmentation masks during training, the masks are replaced with masks that are generated by a separate segmentation model during evaluation.

Since DSFormer does not need to generate segmentation masks itself, a relatively small backbone network is employed. Specifically, we use a ResNet-50 [40] backbone,

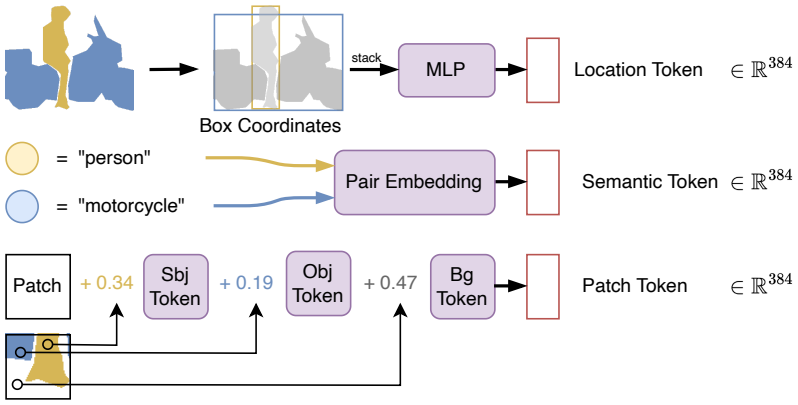


Figure 4.4: Patch tokens are created by encoding the overlap ratio of the subject and the object with the patch. This is achieved by adding a weighted sum of learnable subject, object, and background tokens to the raw feature patch. Location tokens are inferred from a two-layer MLP that processes normalized bounding boxes of the subject and object regions. Semantic tokens are directly obtained from the class labels of subject and object using a learnable embedding that assigns a unique vector to each distinct subject-object class combination.

inherited from Faster R-CNN [107] and pretrained for object detection. A feature pyramid network [71] is used to extract features at four varying resolutions. These tensors are then upscaled to the largest resolution and combined into a single feature tensor using a linear layer. For an input RGB image with a resolution of 640×640, the resulting feature tensor has then dimensions of $160 \times 160 \times 256$. Next, the feature tensor is divided into non-overlapping patches, each with an 8×8 patch size. Each patch is then projected into a token with an embedding dimension of 384. Prior to processing these tokens within the transformer module, subject and object location is encoded.

4.2.3 Subject-Object Encoding

Given the regions of subject and object, DSFormer outputs a predicate classification that describes the relation between the prompted subject and object. Many two-stage approaches [73, 123, 133] leverage bounding boxes retrieved by a first-stage model and extract feature crops using *RoIAlign* [41] or similar techniques. However, for DSFormer, we use an improved method that instead of using the feature crops to indicate the subject and object locations to DSFormer, we retain all information from the backbone and integrate a prompt encoding directly into the patch tokens that were extracted by the backbone. This allows DSFormer to leverage global context information for the final output, including even regions that are outside the subject or object regions. For instance, an image that shows a restaurant is more likely to contain predicates such as “eating” or “drinking”.

Figure 4.4 shows how the prompts are encoded in the patch tokens. The subject-object encoding is a token that is added to the patch token, similar to a positional encoding. The encoding is calculated as a weighted sum of three learnable tokens t_{sbj}, t_{obj}, and $t_{bg} \in \mathbb{R}^{384}$. These tokens encode the presence of the subject, object, and background at a certain patch and are aggregated into a patch token:

$$token = patch + r_{sbj}\frac{t_{sbj}}{||t_{sbj}||} + r_{obj}\frac{t_{obj}}{||t_{obj}||} + (1 - r_{sbj} - r_{obj})\frac{t_{bg}}{||t_{bg}||} \ . \tag{4.4}$$

r_{sbj} represents the ratio of how much of a given patch is covered by the subject mask, with r_{obj} being defined analogously. Since the subject and object masks do not overlap by problem definition, the sum of these ratios is always less than or equal to 1.

An alternative approach to eq. (4.4) is to replace the ratios with binary values (1 if the patch is partially covered by the segmentation mask and 0 otherwise), as defined in eq. (4.4). Experiments show that this approach performs equally well.

$$token = patch + \lceil r_{sbj} \rceil \frac{t_{sbj}}{||t_{sbj}||} + \lceil r_{obj} \rceil \frac{t_{obj}}{||t_{obj}||} \ . \tag{4.5}$$

4.2.4 Transformer Module

DSFormer's core component is a transformer module inspired by a Vision Transformer [27] with 6 blocks. This module receives the patch tokens along with an additional learnable classification token, represented as the filled red box in fig. 4.3. Besides these tokens, the transformer module also accepts optional semantic and location tokens, shown in fig. 4.3 as S and L. More details about the optional tokens can be found in section 4.2.7. Once the classification token has been processed by the transformer module, a two-layer MLP projects it to the desired relation output as indicated by the green arrows in fig. 4.3. This output vector contains a score for each possible predicate and a *no-relation* class in addition.

The *no-relation* predicate serves as a virtual predicate that is not directly defined in the dataset. It is trained to have a low value when there is any relation between a subject and an object, and it should be high if there is no relation estimated. The intuition behind a *no-relation* class is the observation that learning a *no-relation* class is easier than learning accurate individual predicate scores because it is a simpler task. During inference and evaluation, the *no-relation* score is combined with the individual predicate scores, as detailed in section 4.2.9.

4.2.5 Relation Loss

As covered in chapter 5, a reliable evaluation of negative ground truth is impossible. A negative ground truth can arise from different potential sources:

1. The predicate does not apply, and the annotator correctly refrained from using it.

2. An annotator missed adding the label, even though it would have been correct.

3. An annotator preferred to label the relation with a different predicate, even though both predicates would have been correct.

Sources 2 and 3 describe scenarios where faulty negative ground truth is obtained. We observe that 97% of all relations within the PSG dataset are single-label annotations, which suggests that scenario 3 is very common. To cope with this unreliable negative ground truth, our training is designed to prioritize false negatives (which require positive ground truth) over false positives (which require negative ground truth).

To mitigate the influence of potentially inaccurate negative ground truth, we employ a loss function designed to be less sensitive to false positives. For a given predicate class p selected from the set of predicate classes P and a sample index i within a batch of size N, we define $y_{i,p}$ as the ground truth label which is 0 for negative samples and 1 for positive ones. The model's output for predicate p and sample i is represented as $x_{i,p}$. With this, eq. (4.7) defines $l_{i,p}$ as the weighted binary cross entropy loss for a given relation sample.

To reduce the impact of potentially incorrect negative ground truth on the overall training loss, we assign a specific weight w_p to each individual predicate p, defined as the ratio of negative samples to positive samples during training, shown in eq. (4.6). Since there are many more negative samples than positive samples, $l_{i,p}$ will prioritize positive samples. Note that the network will still learn to differentiate between positive and negative samples because of the large number of available negative samples. The loss for a full training batch, shown in eq. (4.8), is the average over all relation samples.

$$w_p \coloneqq \frac{\#\text{ all neg training samples for } p}{\#\text{ all pos training samples for } p} \tag{4.6}$$

$$l_{i,p} \coloneqq -\left(w_p y_{i,p} \cdot log\ \sigma(x_{i,p}) + (1 - y_{i,p}) \cdot log(1 - \sigma(x_{i,p}))\right) \tag{4.7}$$

$$\mathcal{L}_{rel} \coloneqq \frac{1}{N \cdot P} \sum_{i=1}^{N} \sum_{p=1}^{P} l_{i,p} \tag{4.8}$$

The *no-relation* class is treated the same as the predicate classes, however, retrieving positive and negative samples involves an additional step. A relation exhibits a negative *no-relation* value if there exists another predicate label on the relation. To generate samples that have a positive *no-relation* value, we sample subject-object pairs from the dataset that lack existing annotations. These pairs are then assigned a positive *no-relation* ground truth label with all other predicates being labeled as negative. This process is used to create a balanced training set that ensures an equal number of positive and negative *no-relation* samples.

4.2.6 Auxiliary Instance Loss

To give the model additional guidance to efficiently encode its inputs, we apply an auxiliary instance loss in addition to the introduced relation loss. Recall that in every forward pass, DSFormer receives a subject mask and an object mask to predict the relation between the two. To help the model better "understand" these input masks, DSFormer is also required to classify the category of the subject and object.

This is implemented as an additional instance classifier module, as illustrated in fig. 4.3. The instance classifier is a simple two-layer MLP which takes the classification token after the transformer module and converts it to a vector that has twice as many coefficients as there are instance categories. The vector is then split into subject and object output. With this addition, DSFormer now receives a subject and object mask and predicts a subject, object, and relation classification.

For training, we apply a cross-entropy loss to the subject/object outputs and weight each class by the inverse of its frequency in the training set. We take the resulting subject and object classification losses, $\mathcal{L}_{sbj}$ and $\mathcal{L}_{obj}$, and average them: $\mathcal{L}_{inst} = \frac{\mathcal{L}_{sbj}+\mathcal{L}_{obj}}{2}$.

To combine instance loss and relation loss, we use a weighted sum, with empirically determined $\lambda_{rel} = 0.8$ and $\lambda_{inst} = 0.2$.

$$\mathcal{L} = \lambda_{rel} \cdot \mathcal{L}_{rel} + \lambda_{inst} \cdot \mathcal{L}_{inst} \tag{4.9}$$

4.2.7 Additional Input Tokens

Besides the patch tokens and the classification token, DSFormer supports two optional tokens that can further improve the model's performance.

Location Token To improve performance on relations that may benefit from subject/object location information like *on*, *holding*, or *attached to*, we introduce an additional location token. This approach draws inspiration from Ludwig, Harzig, and Lienhart [83] and uses the available segmentation masks to derive a location token on the fly during training and evaluation. The process is depicted in fig. 4.4.

Given the subject and object masks from the first-stage model or the ground truth, we calculate two bounding boxes, represented using the inclusive left (x_1), top (y_1), right (x_2), and bottom (y_2) coordinates. Next, the coordinates are normalized by dividing them by the image width w or height h respectively and shift them to the range of $[-1, +1]$:

$$x_1' = 2 \cdot \frac{x_1}{w} - 1 \tag{4.10}$$

$$x_2' = 2 \cdot \frac{x_2}{w} - 1 \tag{4.11}$$

$$y_1' = 2 \cdot \frac{y_1}{h} - 1 \tag{4.12}$$

$$y_2' = 2 \cdot \frac{y_2}{h} - 1 \tag{4.13}$$

Next, the obtained coordinates are concatenated into one single vector containing all eight values from the two bounding boxes. Finally, a two-layer MLP projects this vector to the token shape that is required by the transformer module. The size of the hidden layer is set to half of the token embedding dimension.

Semantic Information Zellers et al. [140] demonstrated that including subject and object class information improves PredCls[1] performance. We follow a very similar approach to [140] and instruct DSFormer to learn a unique token vector for every combination of subject and object class. Recall that instance class information is always available to DSFormer. During training, we provide the ground truth classes to the model, and during inference, a segmentation model predicts the class labels and provides them to DSFormer. One drawback of encoding semantic information is the limitation to known classes. When these input labels are omitted, DSFormer can also perform zero-shot predictions at the cost of a slight decrease in predictive performance.

4.2.8 Implementation Details

We use the following parameters for DSFormer, unless specified otherwise. The transformer module consists of 6 blocks with an embedding dimension of 384. Before every block, we add a 2D-sine positional encoding as described by Dosovitskiy et al. [27]. In our experiments, we did not notice any substantial differences when replacing 2D-sine with rotary positional embeddings [122]. We train the model with the AdamW optimizer [81] using a batch size of 32, a weight decay of 0.04, and a learning rate of 3.7×10^{-5}. Input images are resized to 640×640 pixels.

To ensure a fair comparison, DSFormer is not pretrained on other tasks. However, if desired, the model can for example be pretrained on segmentation datasets to better teach the model how the segmentation masks are encoded in the patch tokens. In this case, the relation classifier should be disabled and only the instance classifier is trained on the auxiliary subject and object classification task.

4.2.9 Evaluation

During inference, DSFormer processes every subject-object combination within an image. This is computationally feasible because the backbone's output features are only required once per image and can be reused for every subject-object pair.[2] DSFormer thus returns a list of all possible subject-object pairs together with an assigned predicate distribution that includes the *no-relation* predicate.

To calculate *mR@k*, an SGG model must first generate a list of k subject-predicate-object triplets per image that ideally covers the ground truth annotations. DSFormer

[1] Predicate classification. Refer to section 3.2.3 for the definition.
[2] Exact timings are presented in section 4.3.4.

uses the lowest k *no-relation* scores and selects the associated output relations. Next, the output predicate for each of these k relations is determined. Within a relation, DSFormer determines the predicate with the highest score, excluding *no-relation* and sets it to be the predicate of the predicted triplet.

To calculate *mNgR@k*, an SGG model must again output a list of k subject-predicate-object triplets per image. The difference to *mR@k* is that for the same subject-object combination, multiple predicted triplets are allowed. To calculate the ranking, DSFormer combines the estimated predicate score $s_{r,p}$ with the estimated *no-relation* score $s_{r,no}$ for the same relation r into a ranking score $x_{r,p}$, where p is the predicate.

$$x_{r,p} = (1 - \sigma(s_{r,no})) \cdot \sigma(s_{r,p}) \qquad \sigma \text{ is the sigmoid function.} \tag{4.14}$$

Finally, the $x_{r,p}$ scores within an image are used to select the top k predicted relations. To derive the list of subject-predicate-object triplets, the corresponding r and p values are used.

4.3 Experiments

This section presents a re-evaluation of various PSGG approaches, including our newly introduced DSFormer.

Whenever possible, we use released model weights provided by the authors. Otherwise, we train the models following the procedures outlined in their respective publications. We generate predictions following the *SingleMPO* protocol, which involves aggregating segmentation masks that represent the same visual object and removing redundant predicate distributions for identical subject-object pairs, as detailed in section 4.2.1. We evaluate the performance of the different approaches using *mR@k* and *mNgR@k* within the standard PredCls and SGDet tasks, which were introduced in section 3.2. When evaluating with SGDet, an SGG model needs to determine instance masks, instance class labels, and relations. In contrast, models evaluated with PredCls receive ground truth masks and class labels during inference and only have to identify the correct predicate for each relation. Consequently, one-stage methods, which cannot process the provided information, are unable to be evaluated on the PredCls task. The results are presented in table 4.1.

The reported PredCls scores in table 4.1 are very similar to the values originally presented in the PSG paper [137], which uses the flawed *MultiMPO*. This consistency is not surprising because these two-stage methods do not produce overlapping masks and do not return duplicate relation predictions. Hence, they effectively follow *SingleMPO* already. Regarding SGDet on the other hand, our reported results differ significantly from the original work. Figure 4.5 illustrates the big impact on *mR@50* scores when the inappropriate *MultiMPO* protocol is used. One-stage methods typically generate multiple masks per ground truth instance and thus generate duplicate relation predictions which is invalid when evaluating using *R@k* and

	PredCls ↑			SGDet ↑			incorrect
Method	mR@20	mR@50	mNgR@50	mR@20	mR@50	mNgR@50	mR@50†
IMP	11.25	12.72	27.58	8.81	9.78	21.73	7.88
MOTIFS	20.00	21.83	47.98	15.10	16.32	37.96	10.10
GPS-Net	15.46	18.62	33.60	12.35	14.48	27.14	7.49
VCTree	21.19	23.07	50.24	16.29	17.58	39.41	10.20
PSGTR	-	-	-	10.93	11.62	27.57	20.80
PSGFormer	-	-	-	8.20	8.20	21.75	18.30
Pair-Net	-	-	-	18.02	19.64	21.48	28.50
HiLo	-	-	-	17.51	18.33	40.48	37.60
DSFormer	**34.03**	**40.06**	**64.05**	**27.20**	**30.67**	**50.08**	**(50.08)**

Table 4.1: Comparison of different PSGG methods, evaluated on the PSG dataset [137] with *SingleMPO*. Higher scores indicate better performance. One-stage methods cannot be evaluated on PredCls, resulting in missing values in the table. The column denoted with † displays the previously reported *mR@50* scores using *MultiMPO*. Since there are no previously reported *mR@50* scores for DSFormer and the model inherently follows *SingleMPO*, we use a post-processing technique, described in section 4.3.1, to immitate the *MultiMPO* behaviour. As explained in that section, this post-processing step results in the same score as *mNgR@50*.

mR@k. Following mask merging (section 4.2.1), PSGTR averages 4.19 duplicate relation predictions per image which are removed because of *SingleMPO*. Even worse, Pair-Net has 23.44, HiLo has 36.08, and PSGFormer generates as much as 89.36 duplicate predictions per image. Merging these duplicate relations as described in section 4.2.1 uncovers that the *mR@50* scores for all one-stage methods decrease with a maximum decline of 19.3 points compared to previously reported values. Existing two-stage methods remain unaffected as they already align with the principles of *SingleMPO*. Recent SOTA one-stage methods use more up-to-date backbones than the older two-stage methods, putting two-stage methods at an unfair disadvantage because they are still evaluated using older backbones. To compare more fairly, we replace the first-stage model for each two-stage method with the top-performing MaskDINO [63] segmentation model to provide better segmentation masks. This enhancement results in a 7.4 point increase in the *mR@50* score for VCTree and nearly doubles it for GPS-Net. A more detailed analysis regarding the backbone is conducted later in section 4.3.2.

In contrast to recent trends, our findings demonstrate that two-stage methods surpass one-stage methods when evaluated fairly. Our DSFormer model achieves state-of-the-art performance across all reported metrics, exhibiting a +11 point increase in *mR@50* and +10 point increase in *mNgR@50* compared to the previous state-of-the-art on SGDet. Furthermore, DSFormer achieves a *mR@50* score that is than 50% better than one-stage models. In addition to this performance, training DSFormer is remarkably efficient, as illustrated in fig. 4.6.

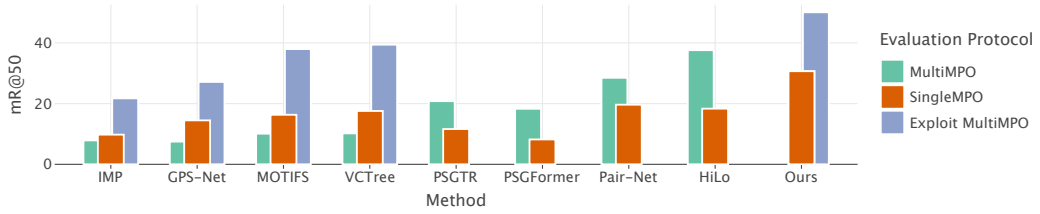


Figure 4.5: *mR@50* scores with different evaluation protocols. (1) *MultiMPO*, the protocol used by prior work which unfairly favours models that return duplicate predictions. (2) *SingleMPO*, our new protocol that corrects these issues. (3) A variant of *MultiMPO* where two-stage methods use improved mask models and exploit *MultiMPO* similar to some one-stage methods. Compared to *MultiMPO*, *mR@50* scores for all one-stage methods decrease when evaluating on *SingleMPO*, with a maximum reduction of 19.3.

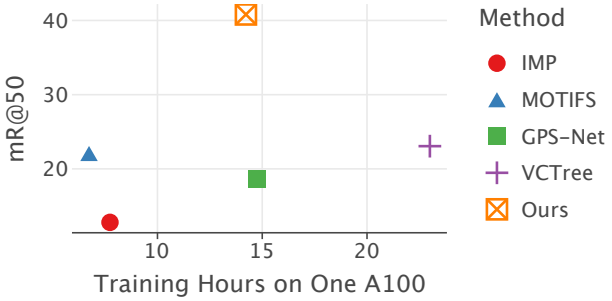


Figure 4.6: Different training times of two-stage methods, all performed on a single A100 GPU. Our method achives significantly better *mR@50* scores while maintaining a comparably low training time.

4.3.1 Evaluating on MultiMPO With DSFormer

To demonstrate how *MultiMPO* can be exploited to gain an unfair advantage over models that adhere to *SingleMPO*, we present an algorithm that converts the *SingleMPO* results from DSFormer to *MultiMPO* post hoc, as shown in algorithm 7. This process modifies the *no-relation* score for each relation and intentionally duplicates predicted relations which would violate *SingleMPO*.

Let n be the number of available predicate classes and $r = (r_0, r_1, \ldots, r_n)$ be a predicted relation from DSFormer, where each $r_i \in [0, 1]$ is a Bernoulli distribution over two classes. r_0 represents DSFormer's *no-relation* score and is not a confidence score for a specific predicate. For each predicate $p \in [1, n]$, a new relation $r^{(p)} = (r_0^{(p)}, r_1^{(p)}, \ldots, r_n^{(p)})$ is derived:

$$r_i^{(p)} = \begin{cases} 1 - ((1 - r_0) \cdot r_p) & \text{if } i = 0 \\ 1 & \text{if } i = p \\ 0 & \text{otherwise} \end{cases} \tag{4.15}$$

Algorithm 7 Convert SingleMPO Output to MultiMPO Output [75]

Input: List R of relation outputs; set of predicate classes P (excluding the no-relation class)
Output: Modified list R' for *MultiMPO*
procedure CONVERT(R, P)
    Initialize empty list R'
    for all relation r in R **do**
        for all predicate p in P **do**
            Initialize new relation r'
            $r'[0] \leftarrow (1 - (1 - r[0]) \cdot r[p])$ ▷ Assign no-relation score
            $r'[p] \leftarrow 1$ ▷ This ensures that the argmax is p
            for all predicate q in $P \setminus \{p\}$ **do**
                $r'[q] \leftarrow 0$
            end for
            Add r' to R'
        end for
    end for
    return R'
end procedure

Using this procedure, we inflate the total number predicted relations and end up with n relation predictions per subject-object pair. Nevertheless, still only k relations are chosen to calculate the final metric. DSFormer uses the *no-relation* score, in this case $r_0^{(p)}$, to determine the top k triplets for recall metrics, as discussed in section 4.2.9.

Using the presented procedure, DSFormer achieves a *mR@50* of 50.08 on the PSG dataset, which is identical to the *mNgR@50* score using *SingleMPO*, as shown in table 4.1. This is to be expected since the described procedure intentionally abuses *MultiMPO* to convert the *mNgR@k* triplets to *mR@k* triplets. For each predicate that is used for *mNgR@k*, the presented algorithm creates a new separate pseudo-prediction where only that predicate is predicted. In *SingleMPO*, this would not work because there are now duplicate subject-object pairs in the model predictions. However, *MultiMPO* is susceptible to this approach.

4.3.2 Influence of First-Stage Models

As we have already stated, PSGG two-stage methods can greatly benefit from the performance of the first-stage segmentation model. As shown in fig. 4.7, the *mR@50* and *mNgR@50* scores achieved on SGDet are directly linked to the *mR@50* and *mNgR@50* scores obtained on PredCls, no matter which segmentation model is used. This suggests that two-stage methods are not tightly coupled with particular first-stage segmentation architectures and can easily be improved by swapping the first stage.

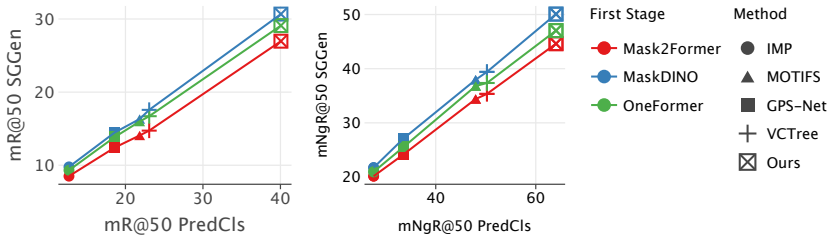


Figure 4.7: Performance comparison between PredCls, where ground truth masks are used (x-axis) and SGDet, where a segmentation model estimates the masks (y-axis). The figure demonstrates strong correlation between the two protocols, indicating that the segmentation model can be swapped. Across all SGG methods, MaskDINO enables the best results.

Figure 4.8 illustrates the impact of the segmentation process on the final scene graph performance. To this end, we compare various top-performing segmentation models [17, 47, 63] and the segmentation outputs from one-stage methods [130, 137, 146]. When integrated with DSFormer, we note that a segmentation model that exhibits high Panoptic Quality (PQ)[3] generally leads to improved *mR@50* scores. The segmentation masks generated by HiLo present an exception since they have a comparatively low PQ but still contribute to DSFormer achieving a competitive *mR@50* of 26.89.

To analyze this phenomenon, we use the $mR@\infty$ metric, introduced in section 3.5.8. $mR@\infty$ simulates a scenario where a perfect SGG model receives segmentation masks and always achieves the highest possible $mR@k$ for any k with the available segmentation masks. Thus, a segmentation model with a high $mR@\infty$ score effectively retrieves masks that are beneficial for maximizing the $mR@k$ metric. As shown in fig. 4.8, $mR@\infty$ and *mR@50* are linked when using DSFormer. However, even though HiLo segmentation masks enable the highest $mR@\infty$ score among all the tested models, they don't lead to the best possible *mR@50*.

The observed behavior can be best explained by interpreting PQ and $mR@\infty$ together. While a high $mR@\infty$ should in theory provide the most relevant masks, we argue that if this comes at the cost of low PQ, the subsequent SGG model cannot be effectively prompted using the low quality segmentation masks. The segmentation models that ultimately enable the highest *mR@50* scores for DSFormer are MaskDINO (30.67), OneFormer (29.10), and Mask2Former (26.97). Among the one-stage methods, HiLo is the only model that provides similarly beneficial segmentation masks, enabling a score of 26.89. Combining these insights, we conclude that the performance of two-stage methods heavily depends on the quality of the provided segmentation masks from the first stage. It is therefore vital to integrate recent segmentation models when comparing against older two-stage methods.

[3]Panoptic Quality [59] is the default metric used for panoptic segmentation. It evaluates both segmentation and recognition quality.

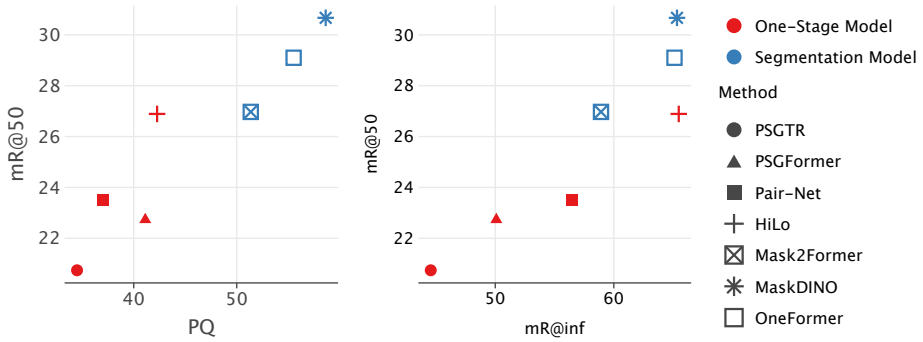


Figure 4.8: This figure illustrates the impact of the first stage on the final scene graph performance, measured in *mR@50*, when using DSFormer as the second stage. The first-stage performance is measured in Panoptic Quality (PQ) and *mR@inf*. *mR@inf* is the best possible mean recall value that a hypothetical perfect second stage model could achieve with the extracted segmentation masks. In addition to pure segmentation models (shown in blue), we also interpret the extracted masks form one-stage methods (shown in red) as a first stage here.

4.3.3 Ablation Study

In this section, we will more closely analyze the various components of DSFormer.

Additional Tokens Table 4.2 presents the contribution of the various components within DSFormer to the final performance. Integrating full mask information and additional semantic and location tokens enhances performance whether applied in combination or individually. When using rectangular masks from bounding boxes instead of more detailed segmentation masks (denoted as $\times$ in the Masks column), the additional semantic token yields a greater performance boost compared to the location token. This is flipped if the full segmentation masks are used (denoted as ✓ in the Masks column), and the location token becomes more impactful. We argue that rectangular masks and location tokens encode overlapping information in distinct ways. Therefore, adding the location token, provides only a marginal improvement (+0.08 *mR@50*). However, when actual segmentation masks and location tokens are used together, they contain complementary information which leads to a substantial performance gain (+6.3 *mR@50*).

Auxiliary Instance Loss Table 4.3 presents the significance of our auxiliary instance loss, introduced in section 4.2.6. If the instance loss is absent, the performance drops from 40.06 *mR@50* to 35.96 *mR@50*, indicating that additional guidance regarding instance information is beneficial. However, enforcing the auxiliary loss too much by a larger weight also harms the performance.

Masks	Location Token	Semantic Token	mR@50	mNgR@50
×	×	×	33.75 ± 0.67	54.47 ± 1.12
×	×	✓	36.52 ± 0.55	56.34 ± 0.96
×	✓	×	33.83 ± 1.15	53.97 ± 1.16
×	✓	✓	38.80 ± 0.79	60.77 ± 1.54
✓	×	×	32.48 ± 0.33	51.80 ± 0.36
✓	×	✓	36.64 ± 0.47	58.36 ± 1.28
✓	✓	×	38.78 ± 0.70	61.75 ± 0.71
✓	✓	✓	**40.06 ± 0.56**	**64.05 ± 1.91**

Table 4.2: Impact of additional tokens on the performance. All performance values are PredCls scores, calculated on the PSG dataset. A × in the Masks column indicates that the enclosing bounding box region instead of the actual segmentation mask is used to prompt the model. The number after the ± symbol represents the standard deviation, calculated by repeating the experiments three times.

Instance Loss Weight	mR@50	mNgR@50
0.00	35.96 ± 2.36	61.04 ± 4.62
0.20	**40.06 ± 0.56**	64.05 ± 1.91
0.50	39.90 ± 0.76	**64.19 ± 1.48**

Table 4.3: Impact of the instance loss weight to the final performance. The number after the ± symbol represents the standard deviation, calculated by repeating the experiments three times.

Embedding Dimension As shown in Table 4.4, an increasing the embedding dimension leads to improved performance. However, the gains on *mR@50* begin to diminish as the embedding dimension grows larger.

Embedding Dimension	mR@50	mNgR@50
192	38.01 ± 1.27	60.67 ± 1.47
256	39.24 ± 1.00	62.88 ± 0.31
384	**40.06 ± 0.56**	64.05 ± 1.91
512	38.90 ± 0.87	**65.50 ± 0.42**

Table 4.4: Impact of the embedding dimension to the final performance. The number after the ± symbol represents the standard deviation, calculated by repeating the experiments three times.

Backbone For a fair comparison, we have limited ourselves to the ResNet-50 backbone [40] used by Faster R-CNN [107], which is used by the other two-stage methods in our comparison. To fully evaluate DSFormer's potential, we re-evaluate using

Backbone	# Parameters	mR@50 ↑
ResNet-50	26.6M	40.06
EoMT-2 Base	93.7M	43.58
EoMT-3 Base	92.4M	**44.13**

Table 4.5: Performance comparison of DSFormer with different backbones. ResNet-50 was finetuned for Faster R-CNN. *#Parameters* denotes the number of parameters of the respective backbone.

Method	# Parameters	Inference
IMP	78M	2.2 min
MOTIFS	108M	1.3 min
GPS-Net	82M	3.4 min
VCTree	104M	4.4 min
PSGTR	44M	4.8 min
PSGFormer	52M	3.9 min
Pair-Net	54M	3.6 min
HiLo	230M	15.2 min
DSFormer	50M	8.4 min

Table 4.6: Comparison of the time taken to process PSG's test set on a single A100 GPU and the number of learnable parameters for various SGG methods.

more up-to-date backbone architectures, namely the EoMT segmentation model [51], which is an adaption of the ViT architecture [27], fine-tuned for panoptic segmentation. To achieve good results, EoMT relies on pretrained weights obtained from DINOv2 [95] and DINOv3 [120]. Both these methods use self-supervised methods to pretrain a ViT model on a very large dataset. To distinguish between the used pretrained weights, we call the model EoMT-2 and EoMT-3 respectively.

We train DSFormer with different backbones and report the results in table 4.5. The models are trained without semantic token. EoMT provides the best backbone which is expected since it received self-supervised pretraining using DINOv2/DINOv3 and additional pretraining for panoptic segmentation.

4.3.4 Inference Speed

To construct a complete scene graph, $n^2 - n$ relations must be classified for n masks.[4] One-stage methods address this challenge by restricting the number of predicted relations to a fixed quantity, typically 100 relations per image. This design choice results in incomplete scene graphs. In contrast, DSFormer is designed to generate complete scene graphs, as we anticipate this to be more beneficial for downstream

[4] Not n^2 because self-relations are excluded.

applications. As demonstrated in table 4.6, this approach is still computationally practical. Our implementation, running on a single NVIDIA A100, can efficiently process approximately 2400 relations within a single forward pass. Additional relations can be processed sequentially without having to recompute the image feature tensor, which drastically reduces execution time. Note that only 0.2% of the images in the PSG dataset require splitting.

4.3.5 Qualitative Examples

Figure 4.9 shows some example scene graph outputs from DSFormer. The model retrieves most relations that are in the ground truth. When people are close together, DSFormer sometimes identifies the relation as "talking to". This might be the case because "talking to" is difficult to annotate in the ground truth based on still images alone. However, in most cases where the model disagrees with the ground truth, it still returns sensible if not better relations. For example, in picture **A** DSFormer recognizes that the bicycle is leaning on the wall and in picture **E**, that another bicycle is in front of the wall. Another example can be seen in picture **A** where the window is annotated as "on the wall", but DSFormer prefers "in the wall".

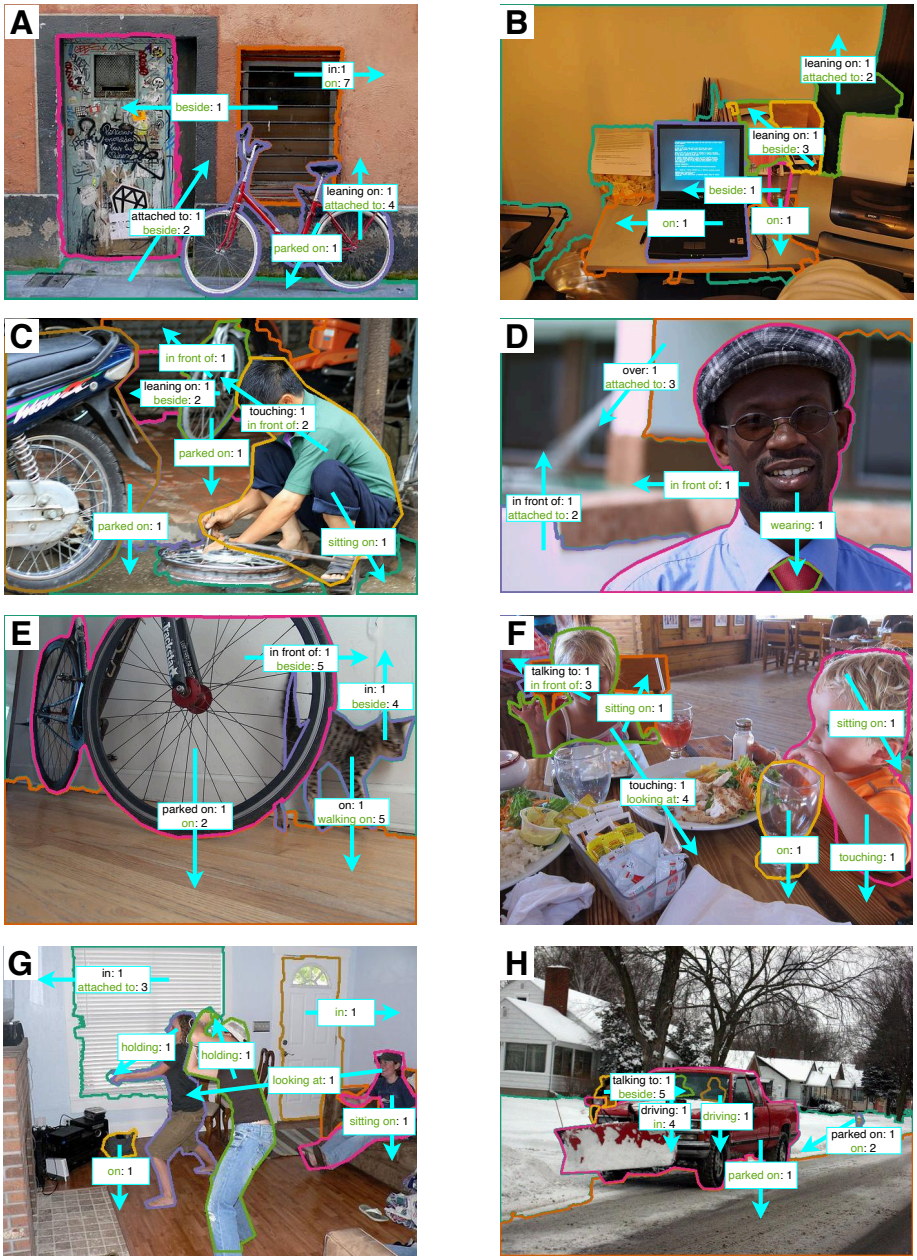


Figure 4.9: Example outputs from DSFormer. The ground truth annotation is represented by blue arrows. For each relation, DSFormer assigns a confidence score to every predicate, which can be used to rank the predicates. The ranks are displayed as numbers after a predicate name. A lower number indicates a higher confidence that the predicate fits compared to those predicates with higher ranks. The maximum possible rank is 56, corresponding to the total number of predicate classes in the PSG dataset. Green text represents the ground truth predicate label for a relation.

4.4 Conclusion

Analyzing recent advances in scene graph generation, we have observed unexpected and undesirable outcomes with the existing evaluation protocol for panoptic scene graph generation. Based on these findings, we have outlined the requirements for a robust evaluation. We believe that our revised *SingleMPO* more accurately reflects the qualities of a well-performing PSGG model and recommend to transition to our new protocol.

Moreover, we have demonstrated that correcting the current shortcomings in the evaluation reveals that existing one-stage methods achieve significantly lower scores than originally reported, while the scores of two-stage methods remain consistent with their original findings. We have introduced DSFormer, a novel two-stage architecture that operates independently of the chosen segmentation model. It can be prompted with subject and object masks that can be derived from any segmentation system. Using a specialized patch encoding procedure, DSFormer outperforms all existing SGG models, achieving a *mR@50* of 30.67 (+11) and a *mNgR@50* of 50.08 (+10).

To further enhance its capabilities, DSFormer could benefit from pretraining or integration with external knowledge [67, 70, 147]. Future research should also explore methods for leveraging information between related relations within an image.

As panoptic segmentation models continue to advance, we believe that two-stage SGG methods hold significant potential for top-performing PSGG methods. Selecting a current and effective first-stage segmentation model is essential to ensure a fair comparison, as SGDet scores will naturally improve with advancements in segmentation methods.

5 Improved Scene Graph Evaluation Using the Haystack Dataset

In this section, we introduce Haystack [79], a new scene graph dataset that we designed and built for an improved and fine-grained evaluation of SGG models. The goal of this dataset is to complement existing scene graph datasets by focusing mostly on rare predicate classes, which cannot be reliably evaluated on existing datasets. Additionally, we provide negative relation annotations which allow for a much more fine-grained analysis using additional metrics.

This chapter is mainly based on the following publication:

Haystack: A Panoptic Scene Graph Dataset to Evaluate Rare Predicate Classes [79], **Julian Lorenz**, Florian Barthel, Daniel Kienzle, and Rainer Lienhart, *2023 IEEE/CVF International Conference on Computer Vision Workshops (ICCVW)*, 2023.

5.1 Introduction

The long-tail problem of scene graph datasets poses a notable hurdle to existing SGG methods [12, 148]. Significant research efforts have been conducted to address this issue, mostly be exploring novel network architectures. While these approaches can lower the performance disparity between common and rare predicate classes, the scarcity of rare predicates within existing datasets is still a limiting factor. For instance, small test sets hinder a reliable evaluation on rare predicates. Moreover, widely used *R@k* metrics primarily provide insights at the image level, often overlooking model performance on the relation level.

In addition to the metrics presented in section 3.5, we introduce new metrics designed to evaluate relations individually and report significant new insights into existing methods. These metrics quantify a model's understanding of specific predicates as well as the influence between predicates prior to their ranking in the final inference output.

To apply the new metrics, reliable relation annotations are essential, including negative annotations. Negative annotations indicate which predicates are *not* part of a certain relation. These explicit negative annotations are absent from prior scene graph datasets, therefore limiting in-depth test set analysis. To overcome this limitation, we create Haystack, a new panoptic scene graph dataset featuring explicit negative annotations with a focus on rare predicate classes. The idea behind

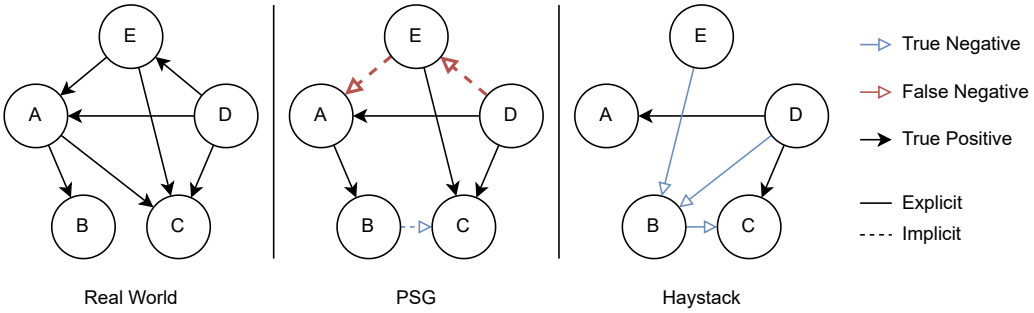


Figure 5.1: A schematic comparison of Haystack's ground truth annotation structure versus prior scene gaph datasets like PSG. Haystack prioritizes more targeted annotations of rare predicates at the expense of complete image annotations. Since it contains explicitely annotated negative annotations, the large amount of unlabeled relations can be ignored. In contrast, PSG is forced to interpret all unlabeled relations as negative annotations.

Haystack is illustrated in fig. 5.1. Even though the dataset is much more sparsely annotated than existing datasets, it is focused on rare relation categories and offers more reliable annotations which are essential for a robust evaluation. Haystack offers the possibility to only rely on explicitly annotated relations while prior datasets have to fall back to heuristics and interpret missing annotations as negative labels.

To allow for a seamless integration with existing training and evaluation pipelines, we base our dataset on images from SA-1B [60]. Relying on set of images that are fully separate from existing scene graph datasets, avoids issues with images that might occur in both the training and test set.

To efficiently create our dataset, we employ an annotation pipeline that is specifically designed to rapidly generate numerous annotations for rare predicate classes. When designing the pipeline, we address two key challenges why scene graph datasets exhibit a long-tail problem:

1. Annotators typically annotate images sequentially without prioritizing images that are likely to contain rare predicates.

2. Annotators often consider the image as a whole and then select the most important relations. In this process they tend to select more basic predicates rather than more informative ones, which become rare predicates.

To mitigate these challenges, we employ a model-assisted annotation pipeline that explores 11 million images from SA-1B [60] to identify and retrieve promising candidates that can then be manually annotated.

The Haystack dataset is compatible with prior scene graph datasets and can be seamlessly integrated into existing evaluation procedures.

The key contributions can be summarized as:

1. We develop an **annotation pipeline** designed to efficiently create scene graph datasets focusing on capturing rare predicate classes. The pipeline leverages model-assisted proposals to identify rare predicates within a large pool of unlabeled images.

2. Using this pipeline, we construct the **Haystack dataset** which contains approximately 25,000 relation annotations across over 11,300 images. Notably, the dataset incorporates explicit negative annotations to facilitate more robust evaluation of rare predicate classes.

3. We introduce **new metrics** that offer a more detailed analysis of SGG model performance on rare predicates, enabling a more comprehensive comparison of existing methods.

5.2 Related Work

In this section, we discuss related work specific to this chapter. We first motivate why we choose PSG as the counterpart for Haystack. Next, we reiterate limitations of existing SGG metrics.

5.2.1 Scene Graph Datasets

As discussed in section 2.4.3, predicate classes follow a long-tail distribution in the PSG dataset and other prior scene graph datasets. However, unlike Visual Genome, PSG contains more reliable annotations. Therefore, we choose PSG to be the training set counterpart for Haystack. Consequently, Haystack shares the same instance classes and predicate classes as PSG.

Creating comprehensive scene graph annotations is extremely challenging. While the PSG dataset has more complete annotations compared to Visual Genome, a significant number of images still contain numerous missing annotations. Figure 5.2 shows examples of such cases.

The introduced annotation pipeline builds upon the predicate classes that are already defined in the PSG dataset, expanding it with new images while focusing on predicate classes that are less frequent in PSG. We recognize that creating exhaustive scene graph datasets is practically infeasible and therefore adopt a different approach with the Haystack dataset. Rather than exclusively focusing on positive annotations, we incorporate negative annotations too, i.e., the dataset includes labels when a predicate class does *not* describe a relation between a certain subject and object. This data offers a more effective way to evaluate individual rare predicate classes, as we will elaborate in section 5.3.3.

5.2.2 Metrics for Scene Graph Generation

Incomplete ground truth annotations in scene graph datasets present a significant hurdle when attempting to apply standard machine learning metrics to the domain

Figure 5.2: This figure shows images from the PSG [137] dataset. The arrows in red indicate relation annotations that do *not* exist in the dataset and were missed by the annotators.

of scene graph generation. To employ metrics like accuracy, both negative and positive labels are essential requirements. However, prior scene graph datasets are only limited to positive relation annotations. Generally, this is not a concern for classification tasks where each data sample typically has a single label. The situation changes for scene graphs, where relations can be associated with zero, one, or even multiple predicate classes. Consequently, the absence of a predicate class in the ground truth does not necessarily indicate that the predicted predicate label is unsuitable for that relation. Indeed, many current scene graph datasets contain images with many missed annotations. Thus, the absence of a predicate class can only be safely considered as an educated guess that a negative annotation should be present.

To address this challenge, a majority of scene graph generation works employ $R@k$ and $mR@k$, which do not require negative ground truth annotations. Both metrics provide insights in a model's ability to rank relevant relations within an image. Please refer to chapter 3 for an introduction to the various metrics.

5.3 Methods

Conventionally, scene graph datasets are annotated sequentially, one image after the other [61, 137]. Using this approach, annotators aim to identify and label as many relations as possible between instances within a single image. For a better annotation quality, annotators are encouraged to choose more detailed predicate classes when applicable. However, this approach has limitations, as it requires annotators to have thorough understanding of all available predicate options to make accurate selections. Thus, we have to modify two core steps in our annotation process to prioritize the identification of rare but more informative predicate classes:

1. To focus on rare predicates, the images are sorted based on an estimated likelihood of obtaining rare predicates. To this end, we employ a model-assisted approach.

2. To address the challenge of annotators needing to consider the entire range of available predicates, we are simplifying the annotation task. Annotators often default to broader predicate classes, such as "on" or "beside" [141] when given a wide pool of choices. Therefore, we structure the process as a binary choice, using a proposal SGG model that presents only relations predicted to belong to a certain predicate class. This allows the annotator to focus on a single predicate, which reduces the potential for classifications with more ambiguous labels.

5.3.1 Annotation Pipeline

Figure 5.3 shows an overview of our annotation pipeline. We begin with a large scale image database and extract objects and their corresponding segmentation masks

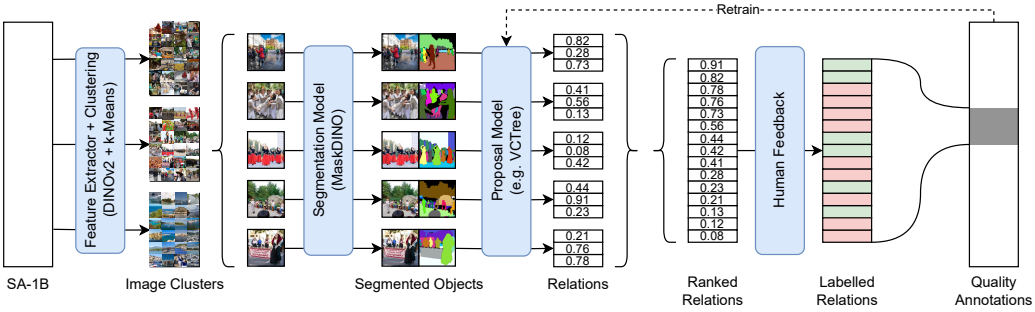


Figure 5.3: Schematic of Haystack's annotation pipeline. The pipeline focuses on identifying rare predicates within a large scale image dataset, e.g., SA-1B. The process begins by clustering the available images to enhance diversity, followed by generation of segmentation masks for each image. The mask format and associated class labels match those of the PSG dataset. Continuing, an SGG model is applied to the obtained images and segmentation masks to predict scores for all potential relationships within the images. These scores are then prioritized and manually annotated. A key difference to prior scene graph datasets are the negative ground truth relation annotations that are obtained during the manual annotation process. Contrary to other datasets, Haystack also contains those annotation types which can be used later for an improved evaluation.

using an off-the-shelf segmentation model like MaskDINO [63]. The retrieved masks then serve as pseudo ground truth data for inference with a pretrained SGG model. The model is used as a proposal method to identify relations that are likely to contain rare predicate classes. Next, human annotators verify the proposed relations and label them as either correct or incorrect relations. In contrast to prior scene graph datasets, we include negative annotations with our dataset which enables novel training and evaluation methods. To ensure dataset diversity, we cluster all images of the source dataset into disjoint groups and sample uniformly from each cluster before processing the images with the introduced pipeline.

Source images Rather than expanding an existing scene graph dataset, we opt to create a completely new dataset that incorporates a fresh set of images. To increase the likelihood of identifying rare predicates, we source all images for Haystack from a large scale dataset, namely SA-1B [60]. This dataset contains over 11 million high quality images spanning diverse domains. Paired with an automatic model-assisted proposal method, we are able to iterate through these images and filter them by their likelihood to contain relevant relations. The proposal method will be introduced more thoroughly in the following sections.

Increase diversity Using a model-assisted approach to propose new relations will introduce a bias towards specific images. A scene graph network will most likely detect more relations on images that are similar to its training set, resulting in re-

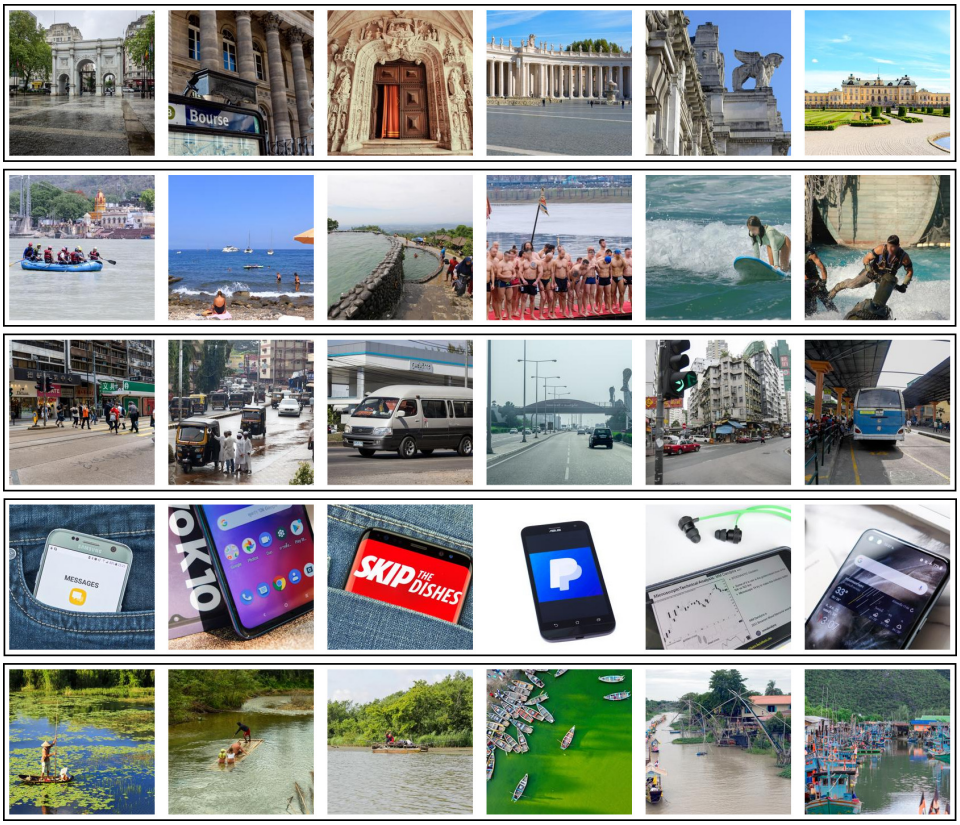


Figure 5.4: Collection of example images from five different clusters, where each row represents examples from a different cluster. The shown clusters are randomly selected out of 50 clusters that were computed using k-Means and features from DINOv2 [95]. Note that some clusters consist of images that are unsuitable for scene graph generation like the fourth cluster, containing only images of smartphones.

duced diversity. To mitigate this risk, we employ a clustering strategy and process the images in groups. For clustering, we use features extracted using a ViT-L [27], pretrained using DINOv2 [95]. DINOv2 is an unsupervised pretraining strategy that enables a model to generate features which are suitable for downstream processing without requiring retraining of the backbone. With the computed image features prepared, we apply k-Means and assign the images to 50 disjoint clusters. Choosing 50 clusters was empirically determined through iterative adjustments on a smaller image subset. 50 clusters represent a practical balance of diverse groupings that still containing enough variation within the groups. Figure 5.4 illustrates multiple example images from the first five clusters. Since not all SA-1B images are appropriate for our scene graph dataset (e.g., portraits or logos), we manually inspect representative images from each cluster to determine whether to exclude the entire cluster.

To generate relation candidates for manual annotation, we combine cluster sampling with model-guided proposals. First, we uniformly sample from the remaining clusters. Then, for each selected cluster, we apply our model-assisted proposal algorithm to rank the most promising candidates. This combination of cluster-based sampling and model-guided proposals ensure the generation of relevant relation candidates from a diverse range of images.

Segmentation masks Although SA-1B provides segmentation masks, they are not compatible with panoptic segmentation masks required for PSG because SA-1B's segmentation masks have no associated class labels and do not represent the categories from the PSG dataset. Manually creating the missing segmentation masks would be inefficient and error-prone. Hence, we leverage MaskDINO [63], trained on the object classes of PSG, to generate predictions for the entire SA-1B dataset. As a foundation model, MaskDINO excels at both object detection and segmentation, achieving state-of-the-art results on COCO instance segmentation. We notice that the extracted segmentation masks are remarkably accurate and suitable for further processing in the introduced pipeline. Failed mask predictions are manually filtered during a later stage of the annotation pipeline.

Predicate renaming We acknowledge that annotators who are unfamiliar with the PSG dataset sometimes struggle to apply the established predicate class definition. The reason being misunderstandings arising from vague predicate class descriptions. This can also be attributed to cultural differences. For example, the PSG dataset was introduced by Chinese researchers who used "playing" and "playing with" as two different predicates. To German annotators, it was not immediately clear what the difference was. To guarantee alignment of our test set with the PSG definitions, we choose to rename the existing predicate classes specifically for our annotators. For example, "playing" is renamed to "engaged in activity using" while "playing with" stays the same. This better clarifies the distinction between the PSG predicates "playing" and "playing with" for our annotators.

To come up with a good one-to-one mapping of predicate class labels for renaming, we use the following approach. For each predicate class label, we obtain a set of images where each image contains at least one relation with the respective predicate. Then, without revealing the original PSG predicate class list, we ask each annotator to assign a predicate label for the presented relations. We allow the annotators to describe the relation freely but concisely. Finally, we verify that the proposed description matches the intended meaning of the predicate class and use the new predicate label as a substitute for the original label. Using this approach, annotators get predicate labels that better fit their understanding of the various predicates labels. Of course for the final dataset, we revert the renamed labels back to their original names.

Figure 5.5: Screenshot of the employed annotation UI. Given a predicate label, in this case "jumping from", annotators are tasked to judge if the proposed relation is either correct or incorrect. On each presented image, the relation is predetermined and a subject and an object are highlighted using distinct colors. Moreover, annotators have the option to mark a segmentation mask as erroneous. In that case, the marked segmentation mask will not be used for future relation proposals. "Absolutely No Relations" is a shortcut if the annotator is sure that none of PSG's predicate classes apply. In the presented example in this figure, an annotator would select "correct".

Annotation interface To build the Haystack dataset, we create a tailored annotation user interface, intended to prevent some common issues with existing scene graph datasets. Theoretically, annotators could label the available images sequentially and select all relations until sufficient data is gathered. However, this approach has two drawbacks:

1. It is highly inefficient to provide extensive annotations for each image. Considering that the pre-processed images contain on average 16 objects which potentially lead to 240 possible relations per image, this approach would result in an unreasonable burden on the annotators.

2. Annotators are less likely to focus on rare predicate classes which we aim to capture.

Our solution to these issues is to fix the predicate class and present potential relation candidates for it. The annotator then has three choices to label the relation, as illustrated in the top button row of fig. 5.5:

1. *Correct*: The annotator selects this option if the predefined predicate matches the given subject-object pair. This selection will result in a positive relation ground truth entry.

2. *Incorrect*: The annotator selects this option if the predicate does not match. This selection will result in a negative relation ground truth entry.

3. *Skip*: If unsure, the annotator is allowed to skip the presented image.

Note that subject-object pairs are multi-label in Haystack. If an annotator chooses that a certain predicate is *incorrect*, it results in a negative relation entry. However, the same subject-object pair might also get an additional positive relation entry if an annotator selects what the *correct* predicate is. In theory, a subject-object pair can have up to 56 relation entries in the Haystack dataset.

In addition, annotators have the option to mark segmentation masks as faulty. This is particularly important since the masks are automatically derived using a segmentation model. If a segmentation mask is marked as faulty, all relation proposals that would use the underlying instance are removed from the pool of proposals.

Model-assisted proposals As already mentioned, we employ an SGG model to propose likely relation candidates. To generate proposals for the Haystack dataset, we use VCTree[1] [123], adapted for PSG [137]. Note that our annotation pipeline is independent of any specific type of SGG model. We train VCTree on the original PSG dataset and run inference on SA-1B to compute all possible relations between all possible subject-object combinations.

[1]Note that DSFormer (chapter 4) was created later.

When using an SGG model for relation proposal, it is essential to make sure if the model outputs contain a dedicated "no relations" output. In that case, proposals are more reliable if the "no relation" output is integrated in the ranking. To do so, we divide the predicted score for the respective predicate by the "no relation" score. This approach ensures that relations with a high "no relation" score are presented less frequently.

After all possible relations are checked using the model, the pipeline assigns a confidence score to each predicate on each relation. To propose rare predicates, we now choose a specific rare predicate class and sort the relations based on the associated predicate confidence score. A human annotator then works through the sorted list of proposed relations. This procedure is repeated for all predicate classes of interest. Note that in many domains which involve model-guided proposals, the most confident samples are not shown to human annotators because they are deemed too easy and as such don't contain any useful information for a dataset. In practice however, we have noticed that VCTree struggles with rare predicate classes and the most confident samples are often not correct. Therefore, we do not set a maximum confidence threshold and choose to include all confidence scores.

Filter nonsense To refine model-assisted relation selection, we filter subject-predicate-object triplets that are unlikely to be useful for our new dataset. For example, annotations like "table-drinking-water" are almost never a correct proposal because tables are unlikely to drink something. We leverage statistics from PSG to count how frequently a subject appears together with a predicate while ignoring the object that participates in the relation. Subject-predicate combinations with one or zero occurrences are considered as potential noise or errors in the annotation process and excluded from the relation proposals. The same principle is applied to predicate-object combinations. While this approach reduces dataset diversity, our proposal algorithm can still suggest novel subject-predicate-object triplets if both subject-predicate and predicate-object pairs are already present in PSG. For instance, PSG contains relations with "dog-eating" and "eating-banana", but not "dog-eating-banana".

5.3.2 Dataset Properties

The Haystack dataset is specifically designed to maximize the number of rare predicates to provide a robust test set for less frequent classes. The dataset is also compatible with existing scene graph datasets such as PSG, allowing an easy combination. For better compatibility with PSG, we reuse its predicate class definitions but limit ourselves to the tail classes.

On a total of over 11,300 images, our collected dataset contains more than 25,000 relation annotations. With our annotation pipeline, annotators were able to identify 9% positive annotations of all proposed annotations for rare predicate classes. Figure 5.6 provides a more comprehensive overview over the fraction of approved relation proposals. Compared to PSG's test set, Haystack contains more positive annotations for rare predicates, as illustrated in fig. 5.7.

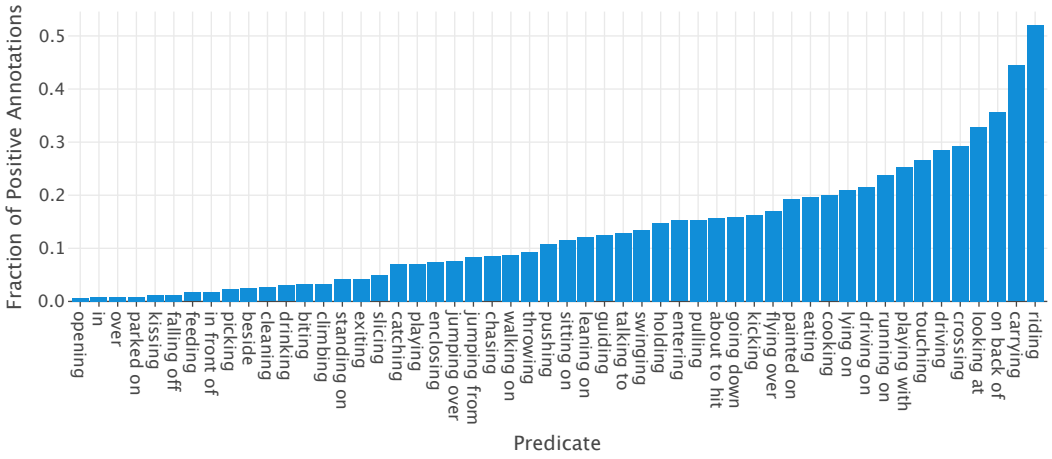


Figure 5.6: Fraction of positive annotations among all annotations in the Haystack dataset. The scores indirectly reflect how difficult it is for the proposal model to retrieve relations that a human annotator approves. For instance, "riding" is effectively suggested by the underlying model.

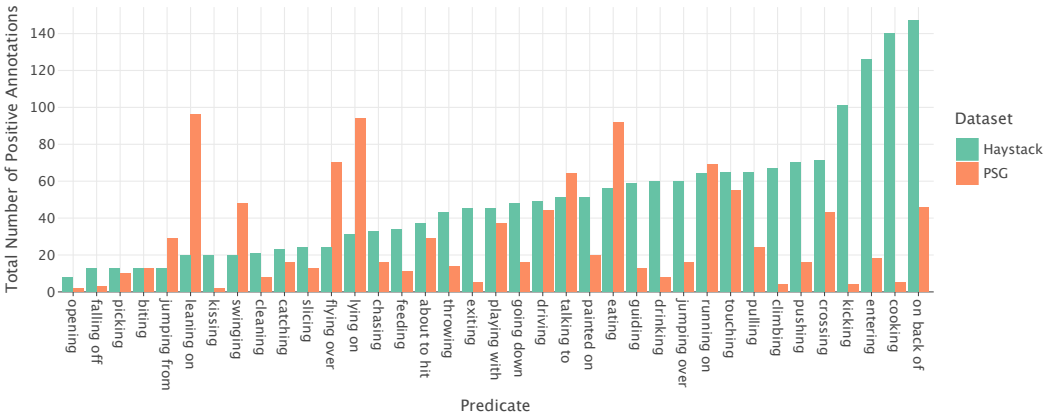


Figure 5.7: Predicate distribution of all positive annotations within Haystack and the PSG test set. Haystack provides a more extensive coverage of rare predicates.

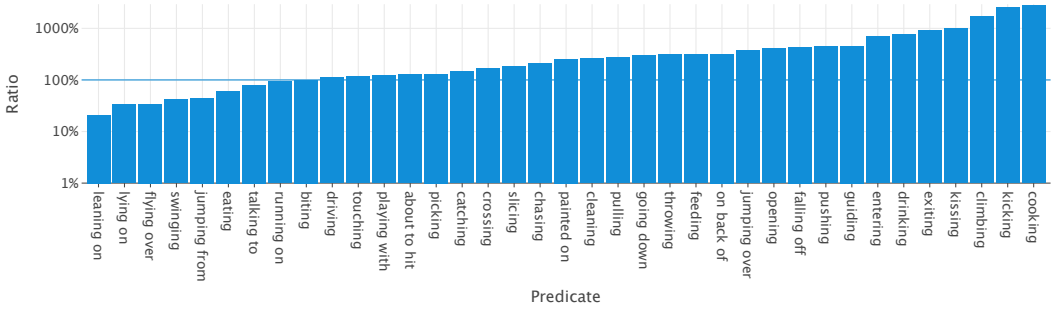


Figure 5.8: Ratio of number of predicates in the Haystack dataset compared to the PSG test set. The plot is log scaled.

Unlike PSG, where annotation is performed on a per-image basis, our annotation process is performed on a per-predicate basis. Therefore, the obtained annotation density per image of Haystack is lower compared to PSG. However, Haystack contains a significantly more rare predicate classes. For instance, Haystack features more than ten times more relations with predicates such as "cooking" or "climbing". Figure 5.8 illustrates how the number of rare predicate annotations is increased compared to PSG.

If an image contains at least one relation annotation, we additionally include the corresponding segmentation masks with SA-1B's image resolution, specifically 1500 pixels along the shorter edge. The relation annotations are stored in a file format that adheres to the same file format as PSG and is 100% compatible. Haystack can be seamlessly integrated into existing PSG-based scene graph pipelines by simply appending our annotations to the existing JSON annotation files.

5.3.3 Evaluation with the Haystack Dataset

Prior work commonly uses $R@k$ and $mR@k$ for evaluation. $R@k$ evaluates relation predictions for each image by calculating the ratio of ground truth annotations that are captured with the top k ranked predictions. Hence, relation predictions are not evaluated individually but effectively compete with each other for the highest ranking positions. As a result, $R@k$ evaluates two distinct model capabilities simultaneously:

1. the model's ability to identify appropriate predicates for various relations and

2. its capacity to rank those predicates by relevance for the final output.

While this metric is suitable for an overall evaluation, this combined assessment lacks the granularity needed to provide in-depth insights into a model's performance. To address this issue, we introduce three separate metrics that are designed to evaluate the two aspects more independently.

A key prerequisite for our proposed metrics is the presence of negative annotations. For prior scene graph datasets these negative annotations are not explicitly

provided. A potential but rather unreliable approach would be to impute them from positive annotations or missing annotations. As detailed in section 5.2.1, such implicit negatives are unreliable. Fortunately, the Haystack dataset provides explicit negative annotations, which allow us to calculate the new metrics without risking noisy ground truth relations.

Predicate ROC-AUC (P-AUC) We define the *P-AUC* (predicate-wise ROC-AUC) as the ROC-AUC computed over individual predicate scores. To calculate *P-AUC* for a specific predicate class p, we first gather all relations possessing either a positive or negative ground truth annotation. Then, we determine the corresponding predictions for each of these relations and only retrieve the scores associated with predicate p. With these confidence scores and their corresponding labels, we can then calculate the ROC-AUC. This metric offers several advantages for scene graph generation: it is invariant to scaling and transformation of the predicate scores, allowing it to evaluate any predicate class regardless of its average confidence. This is particularly valuable because certain predicates like "carrying" or "pushing" are frequently assigned lower confidence scores compared to other predicates. Ultimately, *P-AUC* evaluates a model's ability to determine whether a given predicate class is applicable to a relation, independent of the predictions that are being made for other predicates.

Predicate Dominance Overestimation (PDO) and Predicate Discrimination Disadvantage (PDD) To better investigate the interplay between different predicate scores, we introduce two displacement metrics: *PDO*, which quantifies how much a predicate class overshadows other predicate classes, and *PDD*, which measures how much a certain predicate itself is overshadowed by other predicates. *PDO* and *PDD* are defined in relation to a specific predicate.

For this section, we use a slightly different definition, to define ground truth relations. Let P be the set of all available predicate classes. For a given subject-object pair, we now define $l \in \{missing, positive, negative\}^{|P|}$ as a vector that simultaneously contains ground truth information about every possible predicate class. For a specific predicate $p \in P$, the coefficient l_p is defined as:

$$l_p = \begin{cases} positive & \text{if } p \text{ is the correct predicate class for this relation} \\ negative & \text{if } p \text{ is the wrong predicate class for this relation} \\ missing & \text{if there is no information whether } p \text{ is correct or not} \end{cases} \tag{5.1}$$

In Haystack, l can contain multiple *positive*, *negative*, or *missing* values. For each relation l, the evaluated model ranks the predicate classes *within* the relation, based on the predicted confidence scores. A lower rank corresponds to a higher confidence. This ranking is represented as $r \in [0, |P| - 1]^{|P|} \subset \mathbb{N}^{|P|}$, where r_p determines the rank of predicate p and all coefficients have a unique rank within r.

Let R_p denote the set of relation annotations that contain a positive or negative ground truth annotation for predicate p together with the associated ranking esti-

mation for the whole relation: $R_p = \{(l, r) \,|\, l_p \neq missing\}$. Finally, we define Φ_p as the set containing all relations with positive annotations for predicate p and $T_{p,k}$ as the set of relations which were predicted with a score within the top k predictions. *PDO* and *PDD* are then defined as:

$$|P| = \text{number of predicate classes} \tag{5.2}$$

$$T_{p,k} \coloneqq \{(l, r) \,|\, r_p < k, (l, r) \in R_p\} \subset R_p \tag{5.3}$$

$$\Phi_p \coloneqq \{(l, r) \,|\, l_p = positive, (l, r) \in R_p\} \subset R_p \tag{5.4}$$

$$f(k, p) \coloneqq \begin{cases} 1 & \text{if } |T_{p,k}| = 0 \\ \frac{|T_{p,k} \cap \Phi_p|}{|T_{p,k}|} & \text{otherwise} \end{cases} \tag{5.5}$$

$$PDO_p \coloneqq 1 - \frac{1}{|P| - 1} \sum_{k=1}^{|P|-1} f(k, p) = \text{“1 - precision”} \tag{5.6}$$

$$PDD_p \coloneqq 1 - \frac{1}{|P| - 1} \sum_{k=1}^{|P|-1} \frac{|T_{p,k} \cap \Phi_p|}{|\Phi_p|} = \text{“1 - recall”} \tag{5.7}$$

For *PDO*, we add a special case when $|T_{p,k}| = 0$ to avoid dividing by zero in eq. (5.5). This can occur when a model never predicts a predicate with rank 0. The sums in eqs. (5.6) and (5.7) are limited by $|P| - 1$ because the maximum value for elements in $T_{p,k} \subset R_p$ is $|P| - 1$.

A high *PDD* score indicates a situation where the predicate appears infrequently in the top scores despite being expected to be present. Essentially, it is being displaced by other predicates, as illustrated in the right example in fig. 5.9. Conversely, a high *PDO* score indicates that a predicate is overly represented in the top scores but is not expected there. This signals that the predicate is displacing other predicates, as illustrated in the left example in fig. 5.9. It is important to note that both *PDD* and *PDO* are linked and evaluation should always be performed with both metrics.

Because *PDD* and *PDO* are defined using recall and precision scores, they are robust to imbalanced label distributions. This is critical since a metric that is susceptible to the positive-negative ratio would produce skewed results, as the Haystack dataset exhibits varying negative ratios across different predicate classes.

5.4 Experiments

For our experiments, we evaluate the PredCls models from the PSG paper [137]: MOTIFS [140], GPSNet [73], and VCTree [123]. More specifically, we use the ResNet-101 variants and report our proposed metrics using the Haystack dataset. In addition, we compare the results with our introduced DSFormer (section 4.2.2). It is not possible to evaluate the newly introduced metrics on the PSG test set because the metrics require reliable negative ground truth annotations which are not available for prior scene graph datasets like PSG.

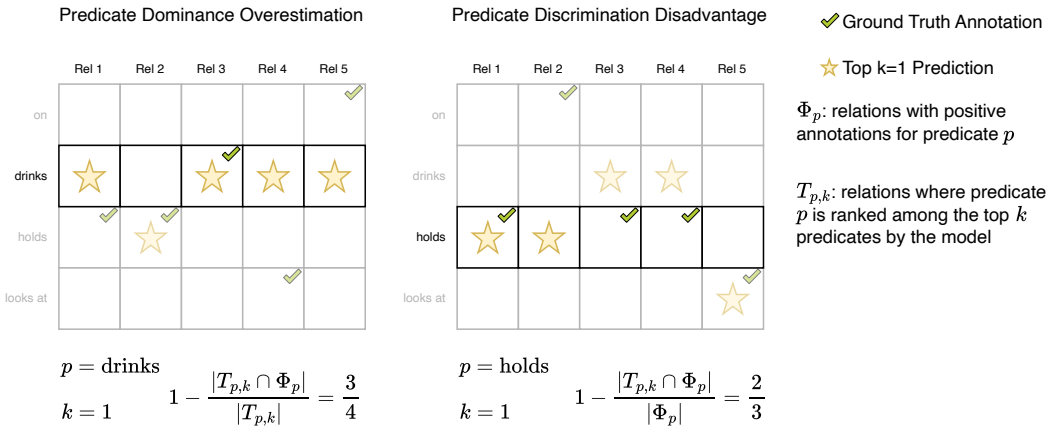


Figure 5.9: Simplified illustration of the PDO (Predicate Dominance Overestimation, eq. (5.6)) and PDD (Predicate Discrimination Disadvantage, eq. (5.7)) metrics. PDO measures how often a predicate incorrectly displaces other predicates. In the left example, "drinks" is predicted four times as the most likely predicate. However it is only correct once, thus wrongfully displacing other predicates in the remaining three cases. PDD measures how often a predicate gets incorrectly displaced by other predicates. In the right example, the predicate "holds" would have been correct three times, but is only predicted a single time as the most likely predicate. In the other two cases, it gets incorrectly displaced by other predicates. These two examples show only the score for $k = 1$. For the full metric, all possible values for k are evaluated and averaged.

	VCTree [123]				GPSNet [73]				MOTIFS [140]				DSFormer			
Predicate	P-AUC	PDD	PDO	R@50	P-AUC	PDD	PDO	R@50	P-AUC	PDD	PDO	R@50	P-AUC	PDD	PDO	R@50
about to hit	0.70	0.26	0.49	0.67	0.73	0.33	0.38	0.70	0.64	0.35	0.56	0.74	0.14	0.75	0.76	0.85
catching	0.46	0.74	0.48	0.00	0.39	0.65	0.64	0.00	0.35	0.87	0.60	0.00	0.71	0.38	0.67	0.07
chasing	0.42	0.72	0.71	0.00	0.40	0.53	0.60	0.00	0.23	0.63	0.63	0.00	0.64	0.04	0.95	0.00
cleaning	0.66	0.77	0.61	0.00	0.67	0.84	0.39	0.00	0.64	0.91	0.67	0.00	0.93	0.42	0.67	0.00
climbing	0.67	0.84	0.58	0.00	0.72	0.94	0.35	0.00	0.68	0.68	0.65	0.00	0.85	0.78	0.48	0.00
cooking	0.62	0.83	0.48	0.00	0.59	0.77	0.39	0.00	0.63	0.84	0.54	0.00	0.78	0.63	0.35	0.80
drinking	0.49	0.50	0.77	0.00	0.49	0.52	0.78	0.00	0.59	0.61	0.76	0.00	0.68	0.23	0.93	0.62
eating	0.90	0.19	0.33	0.02	0.64	0.24	0.56	0.04	0.56	0.23	0.52	0.09	0.74	0.77	0.19	0.66
enclosing	0.99	0.05	0.70	0.03	0.90	0.12	0.62	0.03	0.72	0.11	0.72	0.07	0.11	0.82	0.89	0.26
entering	0.57	0.45	0.55	0.00	0.61	0.45	0.47	0.00	0.52	0.43	0.55	0.00	0.31	0.89	0.84	0.00
exiting	0.49	0.70	0.63	0.00	0.49	0.73	0.55	0.00	0.58	0.42	0.73	0.00	0.64	0.05	0.89	0.20
going down	0.89	0.24	0.59	0.00	0.61	0.38	0.65	0.07	0.38	0.57	0.66	0.07	0.79	0.93	0.11	0.25
guiding	0.53	0.44	0.69	0.00	0.61	0.47	0.69	0.00	0.54	0.72	0.62	0.00	0.70	0.05	0.86	0.15
jumping over	0.51	0.53	0.73	0.00	0.51	0.50	0.71	0.00	0.61	0.61	0.66	0.00	0.87	0.30	0.70	0.31
kicking	0.67	0.85	0.45	0.25	0.59	0.69	0.56	0.00	0.65	0.84	0.62	0.25	0.43	0.13	0.88	0.75
leaning on	0.62	0.15	0.74	0.00	0.78	0.13	0.71	0.00	0.68	0.18	0.77	0.00	0.30	0.52	0.58	0.21
lying on	0.68	0.67	0.35	0.22	0.59	0.40	0.54	0.19	0.34	0.45	0.54	0.26	0.89	0.06	0.77	0.70
on back of	0.48	0.27	0.62	0.00	0.30	0.37	0.53	0.03	0.64	0.27	0.58	0.00	0.84	0.19	0.64	0.04
opening	0.63	0.90	0.51	0.00	0.43	0.92	0.40	0.00	0.60	0.84	0.54	0.00	0.88	0.26	0.91	0.00
painted on	0.66	0.19	0.48	0.02	0.61	0.21	0.52	0.00	0.60	0.24	0.51	0.00	0.20	0.86	0.34	0.60
playing with	0.91	0.44	0.51	0.00	0.80	0.39	0.73	0.00	0.83	0.60	0.42	0.00	0.88	0.73	0.63	0.17
pulling	0.49	0.33	0.44	0.05	0.43	0.27	0.51	0.07	0.43	0.63	0.50	0.05	0.84	0.11	0.41	0.50
pushing	0.51	0.65	0.64	0.00	0.57	0.61	0.49	0.00	0.64	0.70	0.49	0.00	0.78	0.07	0.74	0.00
slicing	0.50	0.45	0.85	0.00	0.50	0.43	0.83	0.00	0.37	0.72	0.76	0.00	0.55	0.62	0.95	0.46
swinging	0.83	0.46	0.23	0.08	0.73	0.62	0.35	0.16	0.76	0.73	0.46	0.16	0.92	0.58	0.35	0.67
throwing	0.51	0.62	0.72	0.00	0.42	0.58	0.63	0.00	0.57	0.58	0.63	0.08	0.36	0.59	0.94	0.67
mean	0.63	0.51	0.57	0.05	0.58	0.50	0.56	0.05	0.57	0.57	0.60	0.07	0.64	0.45	0.67	0.34

Table 5.1: Comparison of different scene graph generation models, evaluated on Haystack for PredCls. For reference, we also include the *R@50* metric, calculated on the PSG test set. Note that for *PDD* and *PDO*, lower scores are preferred while for *P-AUC* and *R@50*, higher scores are better. The last row presents the average across all preceding rows, providing a unified score for the entire dataset.

Table 5.1 presents the metric scores for selected rare predicate classes found within the Haystack dataset alongside the *R@50* scores on the PSG test set for comparison. For many of these rare predicates, *R@50* consistently returns the lowest possible value of 0.0, rendering it largely uninformative. Contrary, our proposed metrics provide differentiated values even for predicate classes that are challenging to predict.

The correlation between *P-AUC* and *R@50* is approximately 0.01 and suggests that the two metrics are indeed evaluating different facets of the model's output. Certain predicates such as "playing with" or "eating" exhibit low *R@50* scores but high *P-AUC* scores. This indicates that a model understands these predicates, but they are infrequently represented among the top 50 predictions within an image. For example, in the case of "playing with", this could occur when multiple individuals are engaged in the same activity or when interactions between other objects within the image are prioritized.

It is to be expected that *PDO* values are low for rare predicate classes, as these predicates typically do not displace other predicates. The highest-ranking predicate concerning *PDO* is "slicing", which is logical given that subjects and objects involved in a "slicing" relation often lack alternative plausible predicates. This type of insight is simply not obtainable from the *R@50* metric.

Generally, DSFormer demonstrates superior performance with rare predicates on the standard *mR@50* metric compared to the other methods when evaluated on PSG. DSFormer scores the highest *P-AUC*, indicating the strongest understanding of rare predicate classes. However, while DSFormer also generates the best *PDD* scores, it performs less favorable on *PDO* than all other methods. This indicates that DSFormer is more eager to apply rare predicates when more common predicates would be appropriate. Most notably, the gap between *P-AUC* scores between DSFormer and VCTree is very small, suggesting that DSFormer achieves its *R@50* advantage not because it has a better "understanding" of the predicate classes but because it can better rank them.

Results on the *P-AUC* metric reveal that existing models are indeed capable of understanding rare predicate classes, such as "playing with". The *R@50* metric, however, provides only limited information, simply indicating that these predicates are outranked by others within the image. The *PDD* and *PDO* metrics on the other hand highlight that this issue originates at the relation level. Therefore, future SGG model architectures should address this by prioritizing improved predicate ranking within individual relations. It appears that existing models already possess a basic understanding of the individual predicate classes.

5.5 Conclusion

In this section, we have introduced the Haystack dataset for in-depth evaluation of scene graph generation methods. We have demonstrated how the annotation pipeline is specifically designed to enhance existing scene graph datasets, particularly concerning rare predicate classes. Our model-guided approach streamlines the

annotation process and enables a quick annotation of rare predicates. Haystack facilitates the development of novel scene graph metrics that are tailored to provide deeper relation-level insights into model predictions. With negative annotations available via Haystack, metrics from other fields in computer vision and statistics can be adapted for use within the scene graph context.

6 Parametric Relations and Proto-Relations

In the previous chapters, we have repeatedly discussed issues with current scene graph datasets. In this chapter, we specifically address the problem that many scene graph datasets contain inaccurate ground truth.

To address this challenge, we introduce CoPa-SG, a novel synthetic dataset designed to provide highly accurate ground truth and comprehensive annotations for relations between all instances within a scene. Furthermore, we introduce two new key concepts: parametric relations and proto-relations. Parametric relations offer a more detailed representation of relations by integrating additional information like angles or distances compared to traditional relation types. Proto-relations capture potential relations that emerge when new instances enter the scene. We use CoPa-SG to evaluate several scene graph generation methods and showcase how the newly introduced relation types can be incorporated into downstream tasks to improve planning and reasoning.

This chapter is mainly based on the following publication:

CoPa-SG: Dense Scene Graphs with Parametric and Proto-Relations [77], **Julian Lorenz**, Mrunmai Phatak, Robin Schön, Katja Ludwig, Nico Hörmann, Annemarie Friedrich, and Rainer Lienhart, *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) Workshops*, 2025.

6.1 Introduction

Prior scene graph datasets are typically created through a manual annotation process that relies on textual descriptions of image regions [61, 137]. This approach often leads to inconsistent and incomplete annotations. Traditional datasets prioritize salient relations which is largely due to the impracticality of manually annotating every relation within an image. Even recent efforts that use vision-language models [131] tend to concentrate on these prominent relations. Consequently, existing datasets remain incomplete and fail to capture many valid relations that are present in a scene. Another challenge in scene graph annotations arises from the use of predicate classes like *behind* or *left* which are inherently dependent on the viewer's perspective. Especially spatial relations can be subject to ambiguous spatial interpretations [111, 112, 125]. Relying on subjective interpretations by annotators contributes to inconsistencies in the annotations, because interpretations can vary.

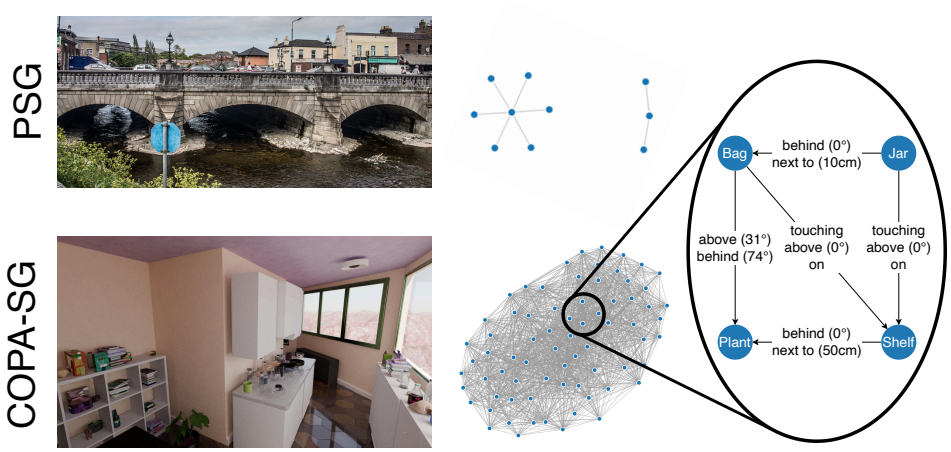


Figure 6.1: Annotation comparison between PSG and CoPa-SG. CoPa-SG has much more comprehensive scene graph annotations compared to traditional scene graph datasets like PSG which primarily focus on salient relations. The magnified graph shows only a selection of the available relations to ensure clarity and readability.

For example, whether an annotator considers a chair to be behind a table can be based the relative distance or requires actual occlusion. Current approaches to mitigate noise in scene graph datasets primarily focus on modifying training protocols [66, 141, 146] rather than directly enhancing the quality of the benchmark datasets themselves.

To address these limitations, we introduce CoPa-SG (**Co**mplete and **Pa**rametric **S**cene **G**raphs), a synthetic scene graph dataset designed to contain highly reliable relation annotations. For this dataset, we draw inspiration from recent research in other computer vision fields that have shown how synthetic data can significantly boost model performance [52, 139]. In this chapter, we present a pipeline that is capable of generating **exhaustive and precise scene graph annotations** for any synthetic dataset with access to the underlying 3D scene. Unlike existing datasets that prioritize salient relations, CoPa-SG provides an exhaustive set of relations for each scene, with an average of 72k relations per scene. An example image is shown in fig. 6.1. Additionally, our annotation process relies on a strict set of deterministic rules which eliminates the subjective interpretation by human annotators. Thus, our dataset enables a more robust training and evaluation of scene graph generation models.

To prevent ambiguously defined predicates, we introduce **parametric relations**, which enrich scene graph representations by additional parameter values. Contrary to traditional predicates, these parametric relations store numerical information like angles or distances alongside the relation. Furthermore, we introduce **proto-relations**, a new technique for representing relations that may potentially arise if new instances are added to the scene at specific locations. For a specific instance

that is already present in the scene, a proto-relation defines the volume that a new instance would need to intersect in order to form a relation with the existing instance. This allows for encoding information like "the area behind the sofa" or "somewhere next to the TV". We argue that this representation provides great potential for agents which rely on an intermediate scene graph to represent contextual knowledge.

CoPa-SG provides the data to evaluate SGG models, especially if they can be employed in downstream applications. In prior work, models were often designed to accommodate the limitations of low-quality ground truth, and thus making it difficult to fully evaluate their potential. We train several scene graph generation models using our high-quality data and evaluate their performance.

Finally, we demonstrate how reasoning can be performed on CoPa-SG scene graphs with small language models. We show that CoPa-SG is expressive and a promising candidate to use in downstream applications.

In this chapter, we discuss the following contributions:

1. We propose a new paradigm to represent relations using **parametric relations**. This paradigm addresses the imprecision inherent in traditional scene graph ground truth labels.
2. We present a novel concept for encoding hypothetical relations within a scene, called **proto-relations**. This relation type can enable support for enhanced reasoning and planning.
3. We develop a pipeline that extracts exhaustive scene graph ground truth from arbitrary 3D scenes.
4. With this pipeline, we construct CoPa-SG, a publicly available synthetic panoptic scene graph dataset containing over 86M relation annotations.
5. We demonstrate how state-of-the-art SGG models can be adapted to detect parametric relations.
6. We provide a comparative analysis of existing models on our dataset.
7. We introduce a simple graph query framework that can be applied to the extracted graphs. This framework could be used for downstream applications and is able to run locally on-device.

The code, dataset, and model weights can be found at `https://lorjul.github.io/copasg`.

6.2 Related Work

With CoPa-SG, we directly address limitations that currently hinder scene graph generation methods. This section provides an analysis of existing datasets and

Dataset	#Relations	Seg	Depth	Normals	Coverage
Visual Genome [61]	2.3M	×	×	×	3.4%
AS-1B [131]	63k	×	×	×	7.1%
PSG [137]	275k	✓	×	×	12.6%
Haystack	6k	✓	×	×	0.2%
3DSSG [128]	544k	✓	✓†	✓†	22.4%
VLA-3D [143]	23M	✓	✓†	✓†	100.0%
CoPa-SG (Ours)	86M	✓	✓	✓	100.0%

Table 6.1: Overview over various scene graph datasets. We denote coverage as the proportion of all possible subject-object pairs that are represented by at least one relation in the ground truth. Entries marked with † denote datasets that have to estimate depth/normals from point clouds. CoPa-SG provides more accurate measurements from 3D meshes.

reveals their shortcomings due to incomplete and inaccurate annotations. Furthermore, we introduce the SGG models used for benchmarking in section 6.5.2.

6.2.1 Scene Graph Datasets

In section 2.4, we have introduced a number of scene graph datasets that are commonly used in scene graph generation. In this section, we introduce additional, more specialized scene graph datasets that are relevant to put CoPa-SG into context. It is important to understand how the datasets were retrieved and what downsides they have as a result. With these downsides in mind, we create CoPa-SG to tackle current limitations in scene graph generation. The most important datasets are summarized in table 6.1.

Visual Genome, discussed in section 2.4.1, was constructed by manually segmenting images into regions of interest and assigning text captions to them. Scene graphs were then extracted from these captions by manually creating relations that match the text caption. As a result, the relations contained in the dataset have a diverse range of labels but are relatively sparsely annotated.

The **AS-1B** [131] dataset approaches the creation process similarly, but is designed for open-world panoptic visual understanding and recognition. Instead of a manual annotation of image regions as it is done for Visual Genome, AS-1B uses large language models (LLMs) and vision language models (VLMs) to annotate the data. A subset of the dataset is verified by human annotators to ensure data quality.

Another dataset that was created similarly is **PSG** [137], which was introduced in section 2.4.2. The dataset makes use of the fact that COCO [72] and Visual Genome share a set of images. Therefore, the authors use the panoptic segmentation masks from COCO and combine them with the relation information from Visual Genome. Additionally, they audit the predicate class definitions of Visual Genome to identify

duplicate predicate labels. This analysis is performed manually and depends on the Visual Genome dataset, thus inheriting similar issues like unreliable annotations and sparse scene graphs that prioritize salient relations.

To tackle these limitations, we create CoPa-SG. Contrary to the discussed datasets, CoPa-SG contains exhaustive scene graphs, unconstrained by predefined image regions. To ensure consistent and reliable annotations, we use a set of well-defined rules to retrieve the scene graphs. Our approach generates high-quality data without requiring user intervention.

The previously introduced **Haystack** dataset (chapter 5) serves a similar purpose as CoPa-SG by focusing on the correctness of annotations. However, they still target different use-cases. While Haystack is a dataset based on real-world images with only a few annotations, CoPa-SG contains synthetic images that are exhaustively annotated. Although Haystack can be used for training when combined with PSG, CoPa-SG enables extensive training of SGG methods on an independent and much larger dataset.

A dataset that follows a similar approach to CoPa-SG is **3DSSG** [128]. Derived from real-world 3D scans of indoor environments [129], 3DSSG contains semi-automatically generated scene graphs. While the authors define a set of rules that are used to extract relations from the 3D scans, the process still requires human verification because the underlying 3D data is noisy. With CoPa-SG, we create a dataset that contains more reliable relation annotations because it depends on less noisy data. Because our dataset is generated fully automatically, we are able to create a much larger dataset.

To facilitate interactive indoor navigation, **VLA-3D** [143] uses existing 3D datasets and enriches them with derived relation annotations. VLA-3D is based on 3D real-world scans from 3RScan [129], Matterport3D [11], Habitat-Matterport3D [105], ARKitScenes [87], ScanNet [20], and custom 3D scenes that were generated in Unity [126]. VLA-3D's approach is quite similar to CoPa-SG, but the authors use a set of heuristics that are based on rotated object bounding boxes. This results in less precise annotations. With CoPa-SG, we use a voxel representation instead of bounding boxes for more fine-grained relation extraction. In addition, we introduce parametric relations which offer greater flexibility than a fixed set of heuristics.

A different approach is presented with **ConceptGraphs** [36]. The authors present a pipeline that fuses multiple views of the same 3D scene into a single scene graph with the help of LLMs and VLMs. With these generated scene graphs, the authors demonstrate various potential downstream applications. ConceptGraphs is not strictly a scene graph dataset but a pipeline to generate graphs using large foundation models. As outlined in the introduction of this thesis, we aim towards reliable scene graph datasets that can be used to train smaller SGG models. As such, relying on LLMs and VLMs is counterproductive. CoPa-SG enables small SGG models to be trained efficiently and used for downstream applications which require on-device processing.

6.2.2 3D Scene Generators

Our dataset creation pipeline that we use for CoPa-SG facilitates a precise and automatic extraction of scene graph annotations from any 3D data like Aria [87] or ProcTHOR [23]. Since CoPa-SG is intended to be a large scale scene graph dataset, we rely on procedurally generated 3D scenes. To this end, we leverage Infinigen [103, 104] together with our automated scene graph annotation pipeline. Contrary to recent neural network based approaches, Infinigen is a rule-based scene generator, which creates realistic, diverse, and complex scenes. Unlike other room layout generators [6, 23, 38, 87] that use premade assets, Infinigen generates each object in the scene procedurally, which significantly expands the diversity of the resulting data.

6.3 CoPa-SG Dataset Construction

In this section, we introduce parametric relations and proto-relations, designed to increase the expressiveness of scene graphs. In addition, we outline the process of creating CoPa-SG.

6.3.1 Parametric Relations

To enhance scene graphs with additional, more detailed information, we introduce *parametric* relations. In addition to standard scene graph relations, which were introduced in section 2.1, parametric relations capture a parameter to provide a more complete description of the associated predicate. The parameters can be distances or angles. For instance, the parametric *adjacent* relation specifies the proximity between two objects, supplemented in CoPa-SG by a numerical distance (e.g., 42 cm). In contrast, traditional scene graphs lack this level of expressiveness because they are limited to binary labels.

Definition Given a fixed scene, we represent instances such as chairs or windows as the set I. The set P is defined as the set of all predicate classes. The predicate class subset $A \subset P$ denotes the set of directional classes like *below* or *in front of*. Complementing A, we have the set of distance-based classes $D \subset P$ which contains classes like *touching* or *adjacent*. Note that by definition, A and D are mutually exclusive ($A \cap D = \emptyset$). Predicate classes without parameters are stored in neither A nor D (e.g., *on*).

In the context of parametric relations, we define a relation by a 6-tuple that stores the following components:

1. Subject instance $sbj \in I$
2. Object instance $obj \in I$
3. Predicate class $pred \in P$

4. Relation parameter $\alpha \in \mathbb{R}$
5. Camera perspective $cam \in \mathbb{R}^{3\times4}$, representing the extrinsic matrix
6. Test direction $\vec{v} \in \mathbb{R}^3$, along which the relation is assessed

A relation can be interpreted as "*sbj* - *pred* - *obj*", for example "chair - touching - bed". CoPa-SG contains the following predicate classes:

- Directional predicate classes: *left*, *right*, *below*, *above*, *behind*, *in front of*
- Distance-based predicate classes: *proximal*, *adjacent*, *close*, *away*
- Other predicate classes: *touching*, *on*

Distance-based relations For distance-based relations, the parameter α represents the shortest distance between *sbj* and *obj*, measured in 3D scene space. Distance-based relations stay the same, regardless of the camera perspective *cam*. To extract these distances, we use a voxel-based approach. In contrast to prior work [143], which relies on rotated 3D bounding boxes, voxels offer a much finer distance estimation. To retrieve the distances, we convert the underlying 3D scene to a voxel representation with 1 cm^3 voxels and efficiently compute the shortest distances between the voxelized objects using K-d trees. Next, we discretize the measured distances into the following intervals (measured in meters):

1. $[0, 0]$: *touching*
2. $(0, 0.3)$: *proximal*
3. $[0.3, 1)$: *adjacent*
4. $[1, 3)$: *close*
5. $[3, \infty)$: *away*

This discretization enables us to train SGG models on CoPa-SG that do not support parameters but only discrete predicate classes. When measuring distances, the granularity is determined by the voxel grid size. While finer voxel grids are certainly possible, they lead to a higher computational burden. Note that CoPa-SG contains distance information about all possible instance combinations.

Directional Relations These relations are a bit different. Here, camera perspective is a crucial factor. While most scene graph datasets are single view datasets, CoPa-SG provides multiple views per scene, meaning that two instances can exhibit entirely different relations depending on the viewpoint. Figure 6.2a provides an example of the camera-dependent *right of* relation. In the figure, the test direction $\vec{v}$ is shown in green. It is orthogonal to the viewing direction of the camera. Similar directional relations are defined similarly, but with a different test direction.

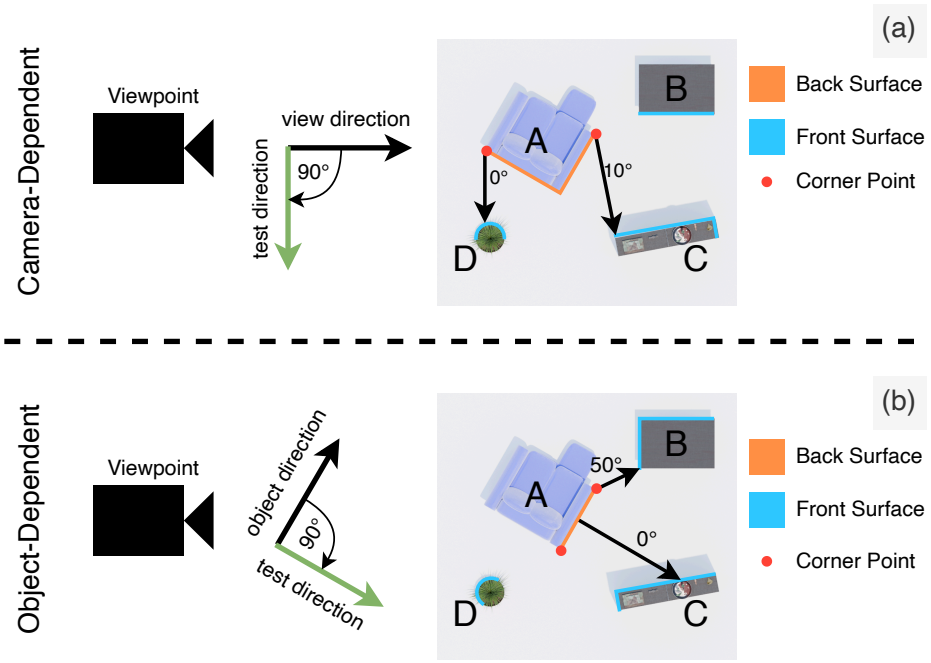


Figure 6.2: Visualization of the directional *right of* relation. The parameter α is defined as the smallest angle between the vector that connects front and back surface (black arrows) and the test direction $\vec{v}$ (green arrows). Depending on whether the relation is camera-dependent or object-dependent, the test direction changes.

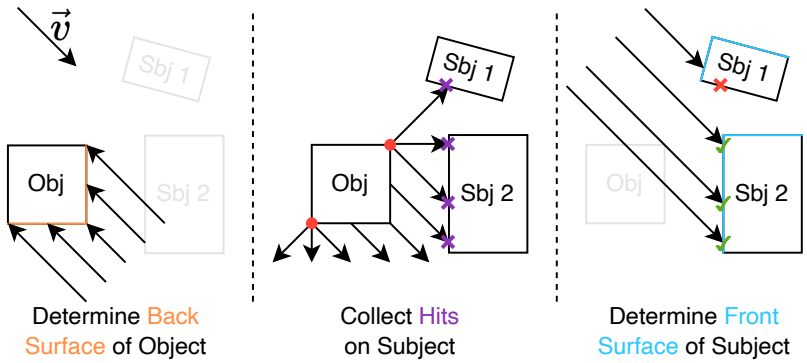


Figure 6.3: Illustration of the ray sweeping process to identify hit locations. In this example, a ray hits *Sbj 1*. However, checking along the $\vec{v}$ direction reveals that the hit location is not on the front surface. The ray is therefore discarded from further processing.

Figure 6.4: Example relation test direction $\vec{v}$ for object-dependent relations. Note that these directions are only defined for selected objects that are listed in table 6.2.

To identify a potential directional relation between *sbj* and *obj*, we measure the angle formed between $\vec{v}$ and the vector that connects front surface of *sbj* with the back surface of *obj*. This process is illustrated in fig. 6.3. The back surface of *obj* is determined by performing a ray casting sweep in the opposite direction of $\vec{v}$. The resulting hit locations define the object's back surface. We use Open3D [145] to perform the ray casts. To identify the front surface of *sbj*, we perform ray casting from the back surface along the direction of $\vec{v}$ to detect ray-intersections with *sbj*. Additionally, at the corner points of the back surface (highlighted with red dots in figs. 6.2 and 6.3), rays are cast that deviate up to 90 degrees from $\vec{v}$ to ensure comprehensive coverage. These corner rays are configured to provide a resolution of 0.05 cm at a distance of 6 m. To confirm that the rays end on the front surface of *sbj* and not somewhere else, we cast an additional set of rays along $\vec{v}$ towards the hit locations. If the new hit locations are too far away from the original hit locations, the original hit was on *sbj*, however not on its front surface. These points are then discarded, as shown in fig. 6.3. Finally, all rays that were not filtered in the front surface check are collected. Among these rays, that with the most similar direction to $\vec{v}$ is selected. The angle between selected ray and $\vec{v}$ is then the parameter α of the extracted relation.

To account for directional relations that are independent of the camera perspective *cam*, we recognize that humans often assign 3D orientation to objects [29]. For example, people consider the "front" of a car to be the at the direction of its typical movement. Consequently, we identify specific object categories generated by Infinigen to which we can assign a "typical" orientation. The comprehensive list of related object categories can be found in table 6.2.

For these objects, we can define individual test directions $\vec{v}$ based on the object's pose, as illustrated in fig. 6.2b. This approach captures how the predicate class can be interpreted when viewed from the object's perspective. Figure 6.4 shows some example test directions for selected objects.

The choice between interpreting relations from the object's perspective or from the camera perspective is a source of ambiguity in human scene relation interpretation. With CoPa-SG, it is clearly defined when which interpretation should

Infinigen Name	Class Label	Directional
Balloon	balloon	×
BarChair	chair	✓
Bathtub	bathtub	×
Bed	bed	✓
BeverageFridge	fridge	✓
Blanket	blanket	×
BlenderRock	rock	×
BookColumn	book	×
BookStack	book	×
Bottle	bottle	×
Bowl	bowl	×
BoxComforter	blanket	×
CeilingLight	light	×
CellShelf	shelf	✓
Chair	chair	✓
Chopsticks	cutlery	×
CoffeeTable	table	×
Comforter	blanket	×
Cup	cup	×
DeskLamp	light	×
Dishwasher	dishwasher	✓
FloorLamp	light	×
Fork	cutlery	×
Fruit	fruit	×
GlassPanelDoor	door	×
Hardware	hardware	×
KitchenCabinet	shelf	✓
Knife	cutlery	×
LargePlantContainer	plant	×
LargeShelf	shelf	✓
LiteDoor	door	×
LouverDoor	door	×
Mattress	mattress	×
Microwave	microwave	✓
Mirror	mirror	✓
Monitor	screen	✓
NatureShelfTrinkets	trinket	×
OfficeChair	chair	✓
Oven	oven	✓
Pan	pan	×
PanelDoor	door	×
Pillar	pillar	×
Pillow	pillow	×
PlantContainer	plant	×
Plate	plate	×
Pot	pot	×
Rug	rug	×
SideTable	table	×
SimpleBookcase	shelf	✓
SimpleDesk	table	×
SingleCabinet	shelf	✓
Sink	sink	✓
Sofa	sofa	✓
Spoon	cutlery	×
StandingSink	sink	✓
TV	screen	✓
TVStand	shelf	✓
TableDining	table	×
Toilet	toilet	✓
Towel	towel	×
Vase	vase	×
WallArt	art	✓
Window	window	✓
Wineglass	wineglass	×

Table 6.2: This table shows which Infinigen objects are mapped to which class labels. Some categories are merged. The "Directional" column denotes which objects support object-dependent relations.

apply. Once $\vec{v}$ is determined, the extraction process follows the same steps as for the camera-dependent directional relations. To distinguish between camera-dependent and object-dependent relations, we categorize the associated predicates as distinct classes, for example, *object front* versus *camera front*.

To predict parametric relations with an SGG model, only the *sbj*, *obj*, *pred*, and α components of a relation are necessary. *cam* and $\vec{v}$ are only important when extracting relations from the 3D scene to form the scene graph dataset.

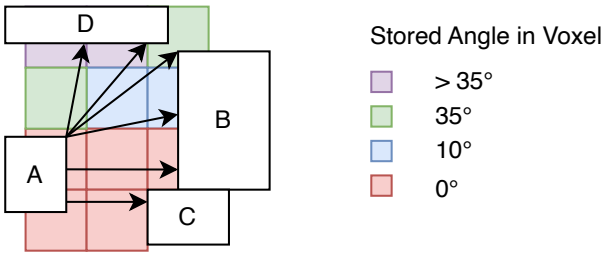


Figure 6.5: This figure illustrates how proto-relations are extracted. First, rays are cast in various directions (black arrows) starting from the anchor object (A). For each voxel that is passed by a ray, the associated ray angle is stored in the voxel. In the end only the smallest recorded angle is kept for each voxel. The ray angle is defined like with parametric relations, see also fig. 6.2.

6.3.2 Proto-Relations

To further enhance the expressiveness of scene graphs, we introduce the concept of proto-relations, which represent hypothetical interactions. We define a proto-relation as a triplet of:

1. Anchor object: an existing instance in the 3D scene.
2. Predicate class label: defines the type of relation, e.g., *on*, *right of*.
3. Proto-volume: a volume within the 3D scene.

A proto-relation essentially states that any new instance in the scene that intersects with the specified volume would automatically fulfill the underlying relation with the anchor object. Proto-relations are valuable for downstream applications, enabling tasks such as answering questions like "Where do I have to place this object so that it will be next to another one?" Figure 6.6 shows an example of such a proto-relation.

The extraction of proto-relations from the underlying 3D scene mirrors the process used for parametric relations. Instead of identifying the object intersected by the rays, we track the space between the ray origin and its hit location. To achieve this, a scene is divided into 1 cm voxels, and we record which voxels are traversed by each ray. Following the same process as in section 6.3.1, we calculate the angular deviation of a ray from the test direction $\vec{v}$ and associate the angle with the respective ray. Next, each voxel that is crossed by a ray gets the associated angle from the ray assigned. If a voxel already has an angle assigned, the smaller angle is chosen, as illustrated in fig. 6.5. This results in a volume of voxels, each associated with an angle. With this data structure, a user can filter the voxels based on their angular deviation from $\vec{v}$, enabling different interpretations of relations, analogous to the approach used for parametric relations.

Proto-relations are compatible with constructive solid geometry (CSG) operations such as union, difference, or intersection. Qualitative examples can be found in

Figure 6.6: Illustration of the *right of* proto-relation for the TV board. Placing an object within the volume highlighted in red establishes a *right of* relation to the TV board with a 0-degree deviation.

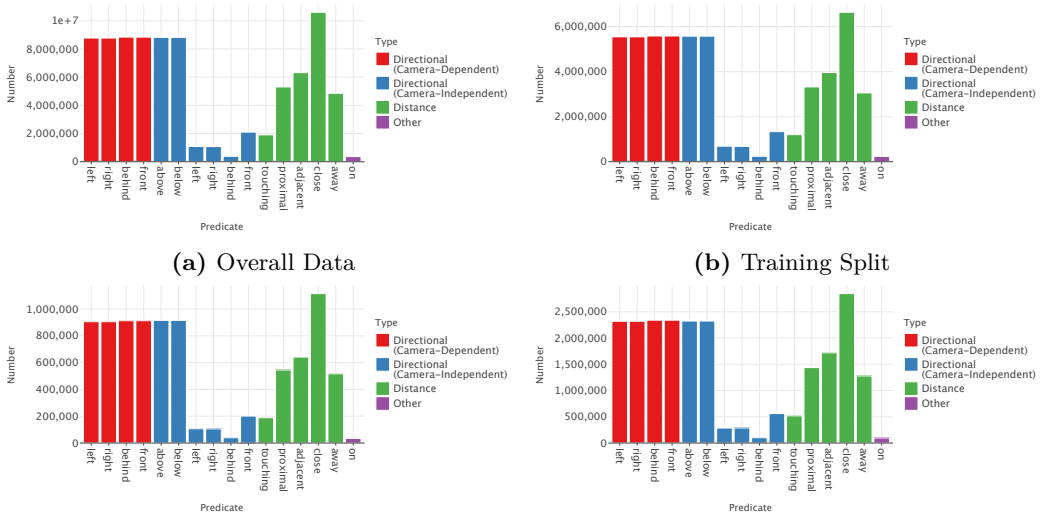
section 6.6. This pairing of proto-relations and CSG operations forms a powerful and expressive base for a variety of downstream applications. CoPa-SG includes proto-relations for each scene in the OpenVDB [89] format which allows efficient data access.

6.3.3 Dataset Statistics

CoPa-SG consists of more than 1.2k unique indoor scenes created from objects that were procedurally created using Infinigen [103], resulting in a comprehensive dataset of over 17 TB of raw ground truth data. We render an average of 30 views per scene, creating a total of 36k images. This setup ensures a sufficient variation while managing computational demands. Figures 6.1 and 6.6 show example renderings. A subset of scenes also receives renderings of up to 200 views per scene to facilitate research on multi-view evaluation scene graphs. A rendered image is accompanied by detailed annotations, including surface normals, depth maps, and segmentation masks.

Because Infinigen lacks class labels for object instances it generates, we create a custom mapping of object names to classes which is listed in table 6.2. It is worth noting that our pipeline is not specific to Infinigen and can be easily generalized to work with any 3D dataset.

Through our comprehensive annotation pipeline, detailed in sections 6.3.1 and 6.3.2, we extract a rich dataset of over 86M relations, averaging 2.4k relations per view and 72k per scene. The distribution of extracted predicate classes is illustrated in fig. 6.7. We provide the corresponding proto-relations for each relation and object instance, as detailed in section 6.3.2.



(a) Overall Data

(b) Training Split

(c) Validation Split

(d) Test Split

Figure 6.7: This figure illustrates the distribution of predicate classes within the dataset.

We provide a significantly more complete annotation of scenes compared to existing datasets, which annotate only 3.4% to 12.6% of possible subject-object pairs (see table 6.1).

We divide the data into training (60%), validation (10%), and test (30%) splits, ensuring that all images from the same scene remain within the same split.

6.4 Scene Graph Generation with Parametric Relations

We now illustrate the extension of SGG models to handle parametric relations. DSFormer from section 4.2.2 is well-suited to analyze scene graph performance on a novel dataset, because of its fully decoupled two-stage design. This approach segregates object detection from relation detection. In contrast to other methods, DSFormer can be trained without a segmentation model, directly using the ground truth segmentation masks. Therefore, we can focus on evaluating scene graph performance instead of additional object detection performance. Unlike many one-stage methods that restrict the number of relations identified per image, two-stage methods typically allow to comprehensively parse all present relations. Another reason for DSFormer is that it supports multi-label predictions for relations between instances. This is a beneficial feature for effectively modeling CoPa-SG, which often includes multiple predicate classes for the same subject-object pairing. For training, we initialize the backbone with DINOv2 [95] pretrained weights and then freeze it, training only the remaining layers.

To adapt an SGG model for parametric relations, we modify the model output as follows. Given a set of n predicate classes that define parametric relations, we first categorize the predicates into angle-based classes $A \subset [1, \ldots, n]$ and distance-based

classes $D \subset [1, \ldots, n]$. To determine if a parametric relation exists between a subject and an object, an SGG model must yield two outputs:

1. A binary flag $\hat{b}$ that indicates whether there is a relation between the respective subject and the object.
2. A relation parameter $\hat{p} \in \mathbb{R}$.

We modify DSFormer's final multi-label prediction layer to simultaneously predict all potential relations for each subject-object pair. To do so, DSFormer outputs a vector of binary flags $\hat{b} \in [0, 1]^n$ and a corresponding vector of parameters $\hat{p} \in \mathbb{R}^n$.

We train the model using ground truth flags $b \in \{0, 1\}^n$ and parameters $p \in \mathbb{R}^n$. The predicted flags $\hat{b}$ are trained using binary cross entropy loss, following the approach outlined in section 4.2.5. To train the parameter outputs $\hat{p}$, we define separate loss terms for distance predictions ($\mathcal{L}_D$) angle predictions ($\mathcal{L}_A$):

$$\mathcal{L}_D(b, p, \hat{p}) = \frac{1}{\sum_{i \in D} f_i} \sum_{i \in D} f_i \cdot \frac{smooth_{L_1}(p_i - \hat{p}_i)}{p_i + 1} \tag{6.1}$$

$$\mathcal{L}_A(b, p, \hat{p}) = \frac{1}{\sum_{i \in A} f_i} \sum_{i \in A} f_i \cdot \left(cos(p_i) - \sigma\left(\hat{p}_i\right)\right)^2 \tag{6.2}$$

In the equations, $smooth_{L_1}$ represents the smooth L1 loss [31]. To prioritize learning closer distances, the distance loss is divided by $p_i + 1$. We apply the sigmoid function to all angle-based parameter outputs before calculating $\mathcal{L}_D$ to constrain the predictions to the range $[0, 1]$. This enables the model to represent the outputs as cosine values for angles within $\left[0, \frac{\pi}{2}\right]$.

6.5 Experimental Results

In this section, we present an evaluation of scene graph generation methods on CoPa-SG, with new metrics and experiments designed to assess performance on parametric relations and multi-view aggregation.

6.5.1 Evaluation Metrics

The non-parametric variant of CoPa-SG is evaluated using (*mNgR@k*) and Mean Average Precision (*mAP*). When parametric relations are present, we use *mAP* alongside Mean Absolute Error for benchmarking.

Non-parametric metrics Because existing scene graph datasets often lack complete relation annotations, SGG models are commonly evaluated using metrics like *R@k* and *mR@k* using $k \in \{20, 50, 100\}$. However, these metrics are unsuitable for CoPa-SG. A hypothetical, perfect model would only achieve an *mR@50* score of 0.077 due to two limitations:

1. *mR@k* is designed for single-label datasets but CoPa-SG is a true multi-label dataset.

2. CoPa-SG's dense annotations means that considering only the top 50 predictions misses many ground truth relations (see section 6.3.3).

Hence, we increase the value of k to 1000 and use *mNgR@k*, a variant of *mR@k* that allows multiple predicates per subject-object pair.

In addition, CoPa-SG enables us to use average precision (AP) for evaluation, contrary to most existing scene graph datasets. In prior work, scene graph evaluation relies on *R@k* metrics because existing datasets only include positive relation annotations [82]. Our dataset, with its exhaustively annotated ground truth relations, overcomes this limitation, and allows us to use metrics like AP per predicate class. Unlike *R@k*, AP considers all predictions within a scene, which yields more robust results, particularly when dealing with imbalanced data. We report the mean average precision (mAP) as the average of these predicate-specific AP scores.

Parametric metrics For parametric relations, evaluation can be split into two separate tasks:

1. Binary classification whether a relation exists or not.

2. Predicting the parameter value if the respective relation exists.

For the relation existence task, we use AP, consistent with our approach for the non-parametric case. For parameter prediction, we calculate the MAE for each predicate class over all ground truth instances that are labeled with that predicate, even if the model does not predict the relation to be present.

6.5.2 Performance with Parameter Thresholds

We introduce CoPa-SG as an extensive benchmark to evaluate SGG models with exhaustive ground truth annotations. To facilitate comparison with prior work, we provide a dataset variant that uses thresholds on relation parameters: angles $\leq 10°$ are considered positive, angles $> 20°$ are considered negative, and intermediate values are ignored. We retrain all compared methods on the same training split of CoPa-SG and evaluate them using both *PredCls* and *SGDet*.

The evaluation results are presented in table 6.3. DSFormer demonstrates the strongest performance on our benchmark, achieving a mAP of 0.307 for *SGDet* and 0.670 for *PredCls*. A noteworthy observation is that MOTIFS [140] outperforms VCTree [123] on *PredCls* when evaluating with CoPa-SG, despite consistently lower results on the Visual Genome and PSG benchmarks. As shown in table 6.3, *mNgR@1000* are not always indicative of mAP performance.

	PredCls ↑		SGDet ↑	
Method	*mAP*	*mNgR@1000*	*mAP*	*mNgR@1000*
MOTIFS [140]	64.1	34.4	27.0	24.4
VCTree [123]	63.9	34.5	27.5	24.9
DSFormer (section 4.2.2)	67.0	45.4	30.7	30.5

Table 6.3: Performance comparison on CoPa-SG, evaluated with parameter thresholds. All methods were retrained specifically for CoPa-SG.

		Camera Independent						Camera Dependent		Average	
Method	Metric	front	above	right	distance	on	touching	front	right	Angle	All
RGB + 1 Block	AP ↑	74.41	82.17	44.43	-	65.44	73.92	77.94	64.65	68.72	68.99
RGB + 1 Block	Parameter Error ↓	22.77°	12.66°	23.30°	1.15 m	-	-	15.59°	17.85°	18.44°	-
RGB + 4 Blocks	AP ↑	87.74	89.32	54.39	-	81.00	82.93	87.81	70.18	77.89	79.05
RGB + 4 Blocks	Parameter Error ↓	17.93°	10.82°	19.71°	1.15 m	-	-	13.17°	15.56°	15.44°	-
Depth + 4 Blocks	AP ↑	86.95	89.74	53.59	-	81.14	83.14	88.94	70.16	77.87	79.09
Depth + 4 Blocks	Parameter Error ↓	17.37°	10.73°	19.32°	1.15 m	-	-	12.54°	14.88°	14.97°	-

Table 6.4: Performance results of our proposed parametric SGG model. To improve readability, we omit results for the *left*/*below*/*behind* predicates as their performance is highly correlated with that of their counterparts.

6.5.3 Estimation of Parametric Relations

Table 6.4 reports the performance of predicting parametric relations using our proposed method. Our evaluation protocol is similar to *SGDet* and provides all compared models with segmentation masks as input without information about the object category. The difference is that we evaluate the parameter predictions in addition. We enhance the original DSFormer by replacing its ResNet-50 backbone with a ViT-S [27] pretrained with DINOv2 [95] to improve performance. All the compared models are trained on the CoPa-SG training set and evaluated on the test set. To investigate the impact of model complexity, we conduct experiments where we set the number of DSFormer's transformer blocks to either 1 or 4. We observe that the smaller model (1 block) trains approximately five times faster but yields lower scores, mostly in the relation existence check with a score of 0.69 *mAP* instead of 0.79. Moreover, we train the models not only on standard RGB images, but we also consider a scenario where our model exclusively receives depth maps as input which are part of CoPa-SG.

The *above* relation appears to be the most reliably predicted across all methods because objects stacked on top of each other usually don't obscure one another. Furthermore, we observe that the object-dependent *right* and *front* relations were more challenging to predict than the camera-dependent versions. To accurately predicate object-dependent relations, a model needs information about the object's pose which adds complexity to the task. Generally, predicting *right* appears to

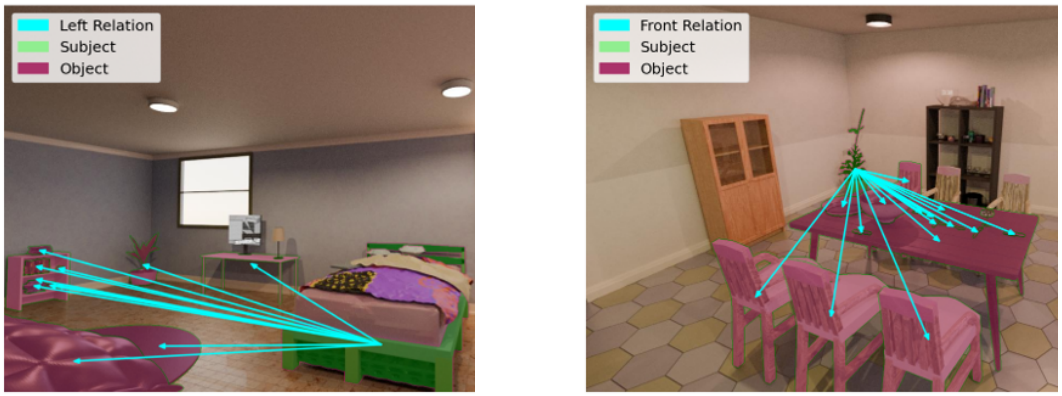


Figure 6.8: Example qualitative results obtained on the CoPa-SG dataset.

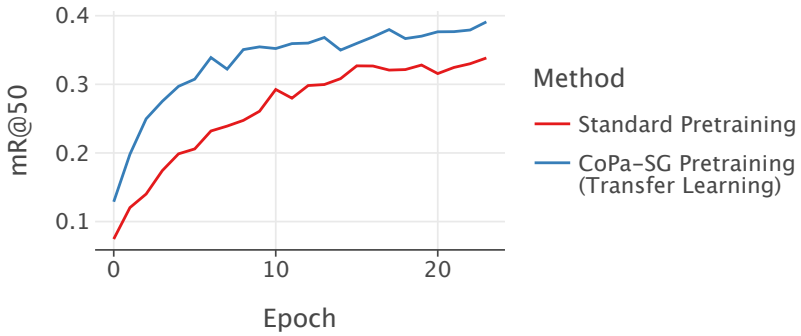


Figure 6.9: Comparison of *mR@50* scores during training, evaluated on the real-world PSG dataset, with and without pretrained weights from CoPa-SG.

be more difficult than *front*. A possible reason for this is that objects which are positioned in front of each other are often closely placed (like items placed in shelves or shelves in front of walls) which requires a model to only focus on a smaller area of the image to infer the relation.

The depth and RGB models perform similarly overall, but the depth model exhibits a consistent improvement in parameter performance. This is expected because the CoPa-SG dataset captures spatial relationships where depth maps provide the model with valuable spatial information.

6.5.4 Using CoPa-SG for Pretraining

CoPa-SG's rich annotations also improve model performance through transfer learning. Our experiments reveal that a significant performance gain on the PSG dataset can be achieved by pretraining on CoPa-SG. Training DSFormer on PSG with and without CoPa-SG pretraining demonstrates a notable performance increase, as illustrated in fig. 6.9.

# Views	behind	front	right	left	above	below	touching	on
1	33.8	65.9	30.2	28.3	86.2	85.6	78.1	76.8
Improvement over single view								
2	+ 0.9	+ 1.6	+ 0.9	+ 1.8	+ 1.2	+ 1.0	+ 1.6	+ 1.6
5	+ 4.8	+ 5.7	+ 4.0	+ 4.6	+ 2.8	+ 2.9	+ 3.4	+ 4.7
10	+ 5.3	+ 6.1	+ 4.5	+ 4.9	+ 3.1	+ 3.1	+ 3.7	+ 5.1
15	+ 5.6	+ 6.2	+ 4.6	+ 5.1	+ 3.2	+ 3.2	+ 3.8	+ 5.3
All	+ 5.6	+ 6.2	+ 4.6	+ 5.1	+ 3.2	+ 3.2	+ 3.8	+ 5.3

Table 6.5: *AP* scores (in %) for object-dependent predicates, showing the improvement achieved by aggregating over multiple views of the same scene. Aggregation is limited to views where both the subject and object are visible. We use the median to aggregate the model outputs.

An explanation for this observation is that most relations have a some spatial component, even those that are primarily semantic. For instance, understanding the *on* relation in CoPa-SG is arguably helpful in determining whether a person is walking on a street. The comparatively large size of CoPa-SG enables pretraining of more complex network architectures.

6.5.5 Multi-View Evaluation

CoPa-SG contains multiple views per scene which enables a multi-view evaluation. We employ an SGG model to predict individual scene graphs per view, focusing on *object*-dependent relations which are independent of the respective camera perspective. To aggregate the individual predictions, we take the median over the model outputs. This differs from the work of Zhang et al. [144], which focuses on associating object recognitions. Instead, we investigate how additional viewpoints can improve relation detection.

Performance improves when we aggregate multiple views as shown in table 6.5. However, it saturates at around 15 views per scene with an AP increase of +6.2. The *front* relation benefits the most from this, likely because it is easier to detect when a view aligns with the object's orientation. The *below* and *above* relations see the least improvement because they already achieve high scores with a single view in the first place.

The ability to use multi-view predictions is particularly beneficial for downstream applications which involve mobile agents that operate and navigate within a scene. While this work concentrates on object-dependent relations, future work should explore methods to reconcile complete scene graphs from various perspectives.

6.6 Reasoning on CoPa-SG

To discuss ideas for future work and demonstrate the potential of CoPa-SG for downstream applications, we provide developer-friendly proof of concept scene graph query frameworks which allow users to easily query information that is encoded within a CoPa-SG scene graph.

6.6.1 Graph Query Framework

We provide a framework to query CoPa-SG scene graphs using natural language, illustrated in fig. 6.10. This framework first converts the predicted scene graph into a Neo4j [92] graph database. We choose a graph database as an intermediate step because they are straightforward to query using query languages such as Cypher [30] for Neo4j. To build the query string, a small language model (SLM, e.g., Qwen 2.5:14b [135] or Phi-4 [1]) receives the user request and transforms it to the query string. To guide the SLM to output a suitable query string, we use a specialized system prompt, found in listing 2. The system prompt instructs the SLM to generate a cypher query and provides additional information about the nodes and relationships in the Neo4j database which are associated with the instances and relations in the predicted scene graph. The text `{{NODES}}` is replaced by the list of observed instance categories from the scene graph, separated by newline characters. Finally, the generated queries are applied to the graph database to retrieve the desired information as structured output. A qualitative example of the whole process is shown in fig. 6.11.

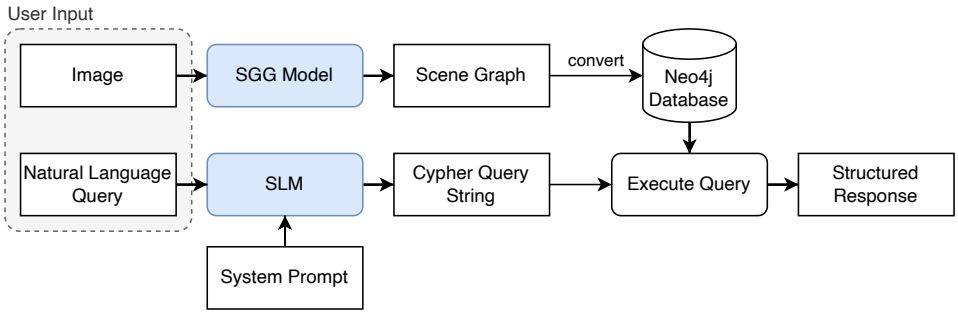


Figure 6.10: Overview of the proposed scene graph query framework. Given an image and a query string, the framework returns a structured response that answers the initial query.

Prompt: How many wine glasses are on the kitchen space?

Cypher:
```
MATCH(wg:Wineglass)-[:ON]->(ks:Kitchenspace) RETURN
count(DISTINCT wg)
```

Result: 5

Figure 6.11: Example query using Qwen 2.5 [135] on CoPa-SG. The SLM generates a Cypher query which is then executed to retrive the response.

Listing 2 System prompt to generate scene graph queries

```
You are an expert in NEO4J and generating CYPHER queries. Help create cypher queries
↪  in valid JSON format {"question": "question provided by the user", "query":
↪  "cypher query"}.
If you are unable to create a query, query should just say "None".

The graph database contains the following nodes:

{{NODES}}

Additionally, the following relationships are present:

// if a is above b. This could also mean floating above.
(a)-[:ABOVE]->(b)
// if a is below b
(a)-[:BELOW]->(b)
// if a is right of b
(a)-[:RIGHT_OF]->(b)
// if a is left of b
(a)-[:LEFT_OF]->(b)
// if a is behind b
(a)-[:BEHIND]->(b)
// if a is in front of b
(a)-[:IN_FRONT_OF]->(b)
// if a is in direct physical contact with b. You can combine this with other
↪  relationships for better results.
(a)-[:TOUCHING]->(b)

There are no other relationships in the graph database!

You are only allowed to make queries using the above information.
```

6.6.2 Proto-volume Query Framework

Furthermore, we provide a similar proof-of-concept query framework for proto-relations, illustrated in fig. 6.12. Given a user prompt, a small language model (SLM) generates a query expression using a specialized subset of Python operators and functions. This query expression is then executed to return a response volume using all available proto-relations. A qualitative example of the whole process is shown in fig. 6.13.

The system prompt for the SLM can be found in listing 3. We instruct the SLM to generate an expression that operates on Python sets using a limited subset of Python, including logical operators and special accessor functions that correspond to a specific proto-relation category. The supported Python subset contains:

- `A & B`: returns the intersection of the two volumes `A` and `B`.
- `A + B` or `A | B`: returns the union of the two volumes `A` and `B`.
- `~A`: returns everything except volume `A`.
- `A - B`: this is equivalent to `A & (~B)`. We have empirically discovered that including this option helps SLMs to come up with better queries.
- `left(A)`, `right(A)`, `above(A)`, `below(A)`, `front(A)`, `behind(A)`, `touching(A)`: returns everything that is left, right, etc. of `A`.

While the system prompt tells the SLM that the accessor functions return sets, the actual implementation uses CSG operations on OpenVDB [89] data structures. We have observed that SLMs return more reliable results if they assume that the expression to generate operates on standard Python sets.

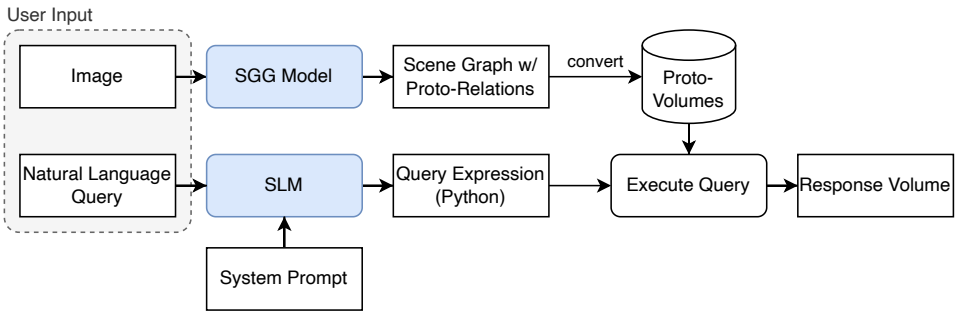


Figure 6.12: Overview of the proposed proto-volume query framework. Given an image and a query string, the framework returns a volume that answers the initial query.

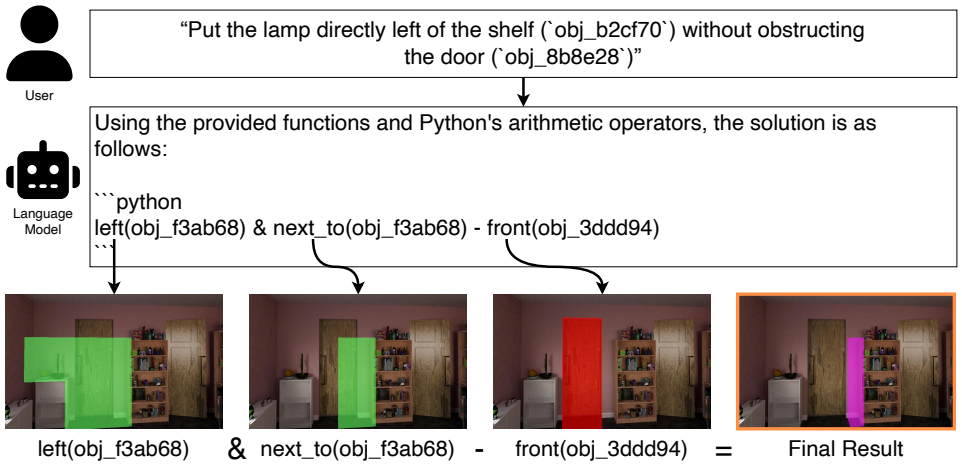


Figure 6.13: Example proto-relatoin query using Phi4 [1]. The SLM generates Python code which is then executed as a set of constructive solid geometry operations to create a response.

Listing 3 System prompt to generate proto-relation queries

```
You are an expert in writing Python code for the following scenario.
Write code that uses the following functions:

``` python
def above(object) -> set:
    """Returns all voxels that are above the specified object. A direct contatct
    ↪  between the voxels and the object is not guaranteed."""
def below(object) -> set:
    """Returns all voxels that are below the specified object"""
def front(object) -> set:
    """Returns all voxels that are in front of the specified object"""
def behind(object) -> set:
    """Returns all voxels that are behind the specified object"""
def left(object) -> set:
    """Returns all voxels that are left of the specified object"""
def right(object) -> set:
    """Returns all voxels that are right of the specified object"""
def touching(object) -> set:
    """Returns all voxels that are directly touching the specified object"""
```

Each of these functions returns a set of voxels. Write python code that uses ONLY
↪ THE ABOVE FUNCTIONS and Python's arithmetic operators: & | ~ - +

The user will provide an instruction. Your task is to find out the requested
↪ location and provide code that will determine the requested location.
In the user prompt, you will see the python variables as obj_XXXXXX, e.g.
↪ obj_c9709f. Use these names as inputs for the functions.
```

6.7 Conclusion

In this chapter, we have presented CoPa-SG, a novel synthetic scene graph dataset with comprehensive relation annotations. In contrast to prior datasets that focus mostly on salient relations, our annotation pipeline generates precise and complete annotations with over 86M relations.

We have introduced two novel relation representations: parametric relations and proto-relations, designed to overcome limitations of traditional scene graphs. **Parametric relations** address the issue of imprecise labels by quantifying relations with an additional parameter (distance or angle). **Proto-relations** extend scene graphs by representing hypothetical relationships using volumes that define where any intersecting object would satisfy the associated relation criterion. This can be used by downstream applications to assist in reasoning and planning tasks. In addition, we have presented a scene graph query framework and have demonstrated potential applications using CoPa-SG.

We argue that scene graphs offer a powerful representation of the world which can be easily parsed by complex systems. The two new relation types that we have introduced provide fundamental building blocks for an efficient use of scene graphs in downstream applications. We believe that expressive and precise data, like that found in CoPa-SG, is a crucial requirement to further progress in this field.

7 Conclusion and Outlook

Coming to the end of this dissertation, we summarize the most notable parts of the dissertation and finish with an outlook over promising new directions in the domain of scene graph generation that build on the presented advances.

7.1 Conclusion

While much recent research on scene graph generation focuses on improving performance through highly crafted network architectures or by leveraging large language models and vision language models, we have focused predominantly on core issues of scene graph generation. As we have outlined in section 1.1, two major challenges in scene graph generation are a fair and correct model evaluation and high quality datasets. Throughout this thesis we have tackled these challenges with various approaches.

We have started by presenting a rigorous definition of scene graph metrics in chapter 3, including mathematical formulations and pseudocode. Next, we have analyzed recent work on SGG methods in chapter 4, where we uncover that a faulty evaluation led to an overestimation of results.

In addition, we have presented two new datasets in this thesis: Haystack (chapter 5) and CoPa-SG (chapter 6). Both datasets tackle the problem of low quality scene graph datasets. While Haystack is based on real world images from SA-1B [60], CoPa-SG is a synthetic dataset that derives accurate scene graph annotations from a virtual environment. Both approaches strive to ensure correct ground truth for improved evaluation and training.

In chapter 4, we have presented a novel SGG model called DSFormer. This neural network architecture has a simple yet effective design that achieves state-of-the-art performance on panoptic scene graph generation. Because of its lean design, adapting the architecture for parametric relations is straightforward, as discussed in section 6.4.

Throughout this thesis, we thoroughly analyzed whether existing scene graph generation evaluations are correct and whether current scene graph datasets contain data structures that are really useful for potential scene graph generation applications, instead of simply trying to improve benchmark scores. Our contributions address the discovered weaknesses of existing work, particularly regarding incorrect evaluation and limited scene graph datasets. Ultimately, we believe these contributions lay a foundation for a more rigorous, reliable but also purpose-driven future for scene graph generation research. We argue that future efforts should always keep the usefulness and applicability of scene graph generation methods in mind.

7.2 Outlook

The research domain of scene graph generation is still a very active research field. Especially the last five years have brought up many interesting new approaches like the panoptic scene graphs [137], 3D scene graphs [128], dynamic panoptic scene graphs [138], open vocabulary approaches [68], or low-latency models [91]. With recent advances in foundation models like AM-RADIO [43, 106] or DINOv3 [120], SGG methods can directly benefit from better scene understanding capabilities of foundation models.

This dissertation creates the foundation for many new research opportunities. A key advantage of scene graphs, as discussed in chapter 1, is that they allow for specialized SGG models. While these models can compete with LLMs and VLMs, they are much more resource-efficient to train and run. Nevertheless, research on real-time capable SGG methods has only received minor attention [91] in the past, even though resource efficiency is one of the main advantages over LLMs and VLMs. With the expressive power of scene graphs, this gives downstream applications on-device reasoning abilities about the observed scene. While most methods focus on sparse scene graphs, we have argued in chapter 6 for dense scene graph predictions. Enabling these predictions to run resource-efficiently is a logical next step to enrich downstream applications. Due to its lightweight design, DSFormer already presents a promising foundation for this development.

In chapter 6, we have introduced CoPa-SG, a synthetic dataset that contains additional information compared to traditional scene graphs via parametric relations and proto-relations. Using a synthetic dataset generator allows us to encode much more detail into a scene graph. However, there is much more information available that can be used to enrich the scene graph representation. Notably, a physics simulator like MuJoCo [124] or Isaac Sim [94] can be used to simulate dynamic scenes which can be recorded in dynamic scene graphs. Dynamic or temporal scene graphs [35, 109, 110, 116, 136, 138] represent how interactions and relations change over time. In contrast to large VLMs, SGG methods can efficiently perform scene graph generation on video data, enabling interesting potential applications, including robots and assistants that navigate through or interact with the scene. Integrating a physics simulator can be used to additionally encode forces and torque within the scene graph, representing physical interactions between instances. This is crucial to build reasoning systems that can understand and predict how instances will behave in a dynamic environment. Fortunately, CoPa-SG uses Infinigen to generate the underlying 3D scene data. Infinigen has built-in support to export the scenes to Isaac Sim which should help to accelerate research in this direction.

When targeting real-world applications, interactions with people is another important factor that should be covered by scene graphs. Recently, we introduced HoIVerse [97], a successor of CoPa-SG which includes posed humans in the scene environment. Future work can build on this and further enrich scene graphs by including information like human joint positions for a better understanding about human-object interactions. For now, HoIVerse contains only static scenes. Another

big step would be to integrate human motion via dynamic scene graphs. However, with human motion, we can also include biomechanical constraints that ensure physically plausible motion. Again, Isaac Sim [94] appears to be a promising candidate since its physics engine can simulate how different joints react to external forces and how they exert forces to other objects. Encoding this information into SGG models may assist the model in better understanding a person's motion and interaction with the environment.

As part of CoPa-SG, we have also introduced proto-relations in section 6.3.2 as an innovative data structure that creates a more expressive scene graphs. While we have focused on reliably and efficiently extracting such proto-relations as well as querying them using small language models, future research should be conducted to design network architectures that can robustly estimate such proto-relations. Being able to predict proto-relations enables new ways to query for navigation and placement problems.

This thesis is focused around tackling fundamental issues in the field of scene graph generation. As such, we have introduced several datasets and methods. During the design of these datasets and methods, we have strived to approach the presented problems from an application view. However, since we have focused on the core issues, we did not investigate potential downstream applications of scene graph generation. With CoPa-SG, we now have an easy to customize data generator that can be configured to include a rich set of scene graph ground truth. Future work should try to develop scene graph generation techniques in conjunction with specific application goals [90] such as complex reasoning [44, 48, 74], robotic navigation [13, 58, 121], or digital twins [57, 101]. As previously discussed, one of the core strengths of scene graphs lies in their ability to be specialized for a particular task. Thus, an application-driven approach is crucial to directly address the application requirements. Fortunately, CoPa-SG provides a flexible platform to easily generate targeted training datasets.

More than ever, scene graph generation is a fascinating research area with many unsolved challenges. Especially as a potential resource-efficient counterpart to recent large vision language models, SGG methods still have many underexplored use-cases and potentials.

List of Figures

List of Tables

Bibliography

[1] Marah Abdin et al. *Phi-4 Technical Report*. 2024. arXiv: 2412.08905 [cs.CL]. URL: https://arxiv.org/abs/2412.08905.

[2] Adobe Systems. *TIFF (Tagged Image File Format) Specification*. Adobe Systems. 1992. URL: https://web.archive.org/web/20180810205359/https://www.adobe.io/content/udp/en/open/standards/TIFF/_jcr_content/contentbody/download/file.res/TIFF6.pdf.

[3] Aniket Agarwal, Ayush Mangal, and Vipul. *Visual Relationship Detection using Scene Graphs: A Survey*. 2020. arXiv: 2005.08045 [cs.CV].

[4] Stanislaw Antol et al. "VQA: Visual Question Answering". In: *2015 IEEE International Conference on Computer Vision (ICCV)*. 2015, pp. 2425–2433. DOI: 10.1109/ICCV.2015.279.

[5] Iro Armeni et al. "3D Scene Graph: A Structure for Unified Semantics, 3D Space, and Camera". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2019, pp. 5664–5673.

[6] Armen Avetisyan et al. "SceneScript: Reconstructing Scenes with an Autoregressive Structured Language Model". In: *Computer Vision – ECCV 2024: 18th European Conference, Milan, Italy, September 29–October 4, 2024, Proceedings, Part LXI*. Milan, Italy: Springer-Verlag, 2024, 247–263. ISBN: 978-3-031-73029-0. DOI: 10.1007/978-3-031-73030-6_14. URL: https://doi.org/10.1007/978-3-031-73030-6_14.

[7] Piotr Bojanowski et al. "Enriching Word Vectors with Subword Information". In: *Transactions of the Association for Computational Linguistics* 5 (2017). Ed. by Lillian Lee, Mark Johnson, and Kristina Toutanova, pp. 135–146. DOI: 10.1162/tacl_a_00051. URL: https://aclanthology.org/Q17-1010/.

[8] Luuk H. Boulogne et al. "The STOIC2021 COVID-19 AI challenge: Applying reusable training methodologies to private data". In: *Medical Image Analysis* 97 (2024), p. 103230. ISSN: 1361-8415. DOI: https://doi.org/10.1016/j.media.2024.103230. URL: https://www.sciencedirect.com/science/article/pii/S1361841524001555.

[9] Nicolas Carion et al. "End-to-End Object Detection with Transformers". In: *Computer Vision – ECCV 2020*. Ed. by Andrea Vedaldi et al. Cham: Springer International Publishing, 2020, pp. 213–229. ISBN: 978-3-030-58452-8.

[10] Mathilde Caron et al. "Emerging Properties in Self-Supervised Vision Transformers". In: *Proceedings of the International Conference on Computer Vision (ICCV)*. 2021.

[11] Angel Chang et al. “Matterport3D: Learning from RGB-D Data in Indoor Environments”. In: *2017 International Conference on 3D Vision (3DV)*. 2017, pp. 667–676. DOI: 10.1109/3DV.2017.00081.

[12] Xiaojun Chang et al. “A Comprehensive Survey of Scene Graphs: Generation and Application”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 45.1 (2023), pp. 1–26. DOI: 10.1109/TPAMI.2021.3137605.

[13] Devendra Singh Chaplot et al. “Neural Topological SLAM for Visual Navigation”. In: *CVPR*. 2020.

[14] Lianggangxu Chen et al. “Multi-Prototype Space Learning for Commonsense-Based Scene Graph Generation”. In: *Proceedings of the AAAI Conference on Artificial Intelligence* 38.2 (2024), pp. 1129–1137. DOI: 10.1609/aaai.v38i2.27874. URL: https://ojs.aaai.org/index.php/AAAI/article/view/27874.

[15] Tianshui Chen et al. “Knowledge-Embedded Routing Network for Scene Graph Generation”. In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019, pp. 6156–6164. DOI: 10.1109/CVPR.2019.00632.

[16] Bowen Cheng, Alexander G. Schwing, and Alexander Kirillov. “Per-Pixel Classification is Not All You Need for Semantic Segmentation”. In: 2021.

[17] Bowen Cheng et al. “Masked-attention Mask Transformer for Universal Image Segmentation”. In: *arXiv* (2021).

[18] Jun Cheng et al. “Visual Relationship Detection: A Survey”. In: *IEEE Transactions on Cybernetics* 52.8 (2022), pp. 8453–8466. DOI: 10.1109/TCYB.2022.3142013.

[19] James H. Clark. “Hierarchical geometric models for visible surface algorithms”. In: *Commun. ACM* 19.10 (Oct. 1976), 547–554. ISSN: 0001-0782. DOI: 10.1145/360349.360354. URL: https://doi.org/10.1145/360349.360354.

[20] Angela Dai et al. “Scannet: Richly-annotated 3d reconstructions of indoor scenes”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 5828–5839.

[21] *Dangerous Pickles — Malicious Python Serialization*. URL: https://intoli.com/blog/dangerous-pickles/ (visited on 06/11/2025).

[22] Matt Deitke et al. “Molmo and PixMo: Open Weights and Open Data for State-of-the-Art Vision-Language Models”. In: *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2025, pp. 91–104. DOI: 10.1109/CVPR52734.2025.00018.

[23] Matt Deitke et al. "ProcTHOR: large-scale embodied AI using procedural generation". In: *Proceedings of the 36th International Conference on Neural Information Processing Systems*. NIPS '22. New Orleans, LA, USA: Curran Associates Inc., 2022. ISBN: 9781713871088.

[24] Alakh Desai et al. "Learning of Visual Relations: The Devil is in the Tails". In: *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*. 2021, pp. 15384–15393. DOI: 10.1109/ICCV48922.2021.01512.

[25] L. Peter Deutsch. *DEFLATE Compressed Data Format Specification version 1.3*. RFC 1951. May 1996. DOI: 10.17487/RFC1951. URL: https://www.rfc-editor.org/info/rfc1951.

[26] Xingning Dong et al. "Stacked Hybrid-Attention and Group Collaborative Learning for Unbiased Scene Graph Generation". In: *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2022, pp. 19405–19414. DOI: 10.1109/CVPR52688.2022.01882.

[27] Alexey Dosovitskiy et al. "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale". In: *International Conference on Learning Representations*. 2021. URL: https://openreview.net/forum?id=YicbFdNTTy.

[28] ECMA International. *ECMA-404: The JSON Data Interchange Format*. ECMA International. 2017. URL: https://www.ecma-international.org/publications/standards/Ecma-404.htm.

[29] Fang Fang and Sheng He. "Viewer-Centered Object Representation in the Human Visual System Revealed by Viewpoint Aftereffects". In: *Neuron* 45.5 (2005), pp. 793–800. ISSN: 0896-6273. DOI: https://doi.org/10.1016/j.neuron.2005.01.037. URL: https://www.sciencedirect.com/science/article/pii/S0896627305000759.

[30] Nadime Francis et al. "Cypher: An Evolving Query Language for Property Graphs". In: *SIGMOD'18 Proceedings of the 2018 International Conference on Management of Data*. Houston, United States: ACM Press, June 2018, p. 1433. DOI: 10.1145/3183713.3190657. URL: https://hal.science/hal-01803524.

[31] Ross Girshick. "Fast R-CNN". In: *2015 IEEE International Conference on Computer Vision (ICCV)*. 2015, pp. 1440–1448. DOI: 10.1109/ICCV.2015.169.

[32] Ross Girshick et al. "Region-Based Convolutional Networks for Accurate Object Detection and Segmentation". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 38.1 (2016), pp. 142–158. DOI: 10.1109/TPAMI.2015.2437384.

[33] Christoph Gohlke. *cgohlke/imagecodecs: v2024.1.1*. Version v2024.1.1. Dec. 2023. DOI: 10.5281/zenodo.10445939. URL: https://doi.org/10.5281/zenodo.10445939.

[34] Christoph Gohlke. *cgohlke/tifffile: v2024.2.12*. Version v2024.2.12. Feb. 2024. DOI: 10.5281/zenodo.10651760. URL: https://doi.org/10.5281/zenodo.10651760.

[35] Elias Greve et al. "Collaborative Dynamic 3D Scene Graphs for Automated Driving". In: (2024), pp. 11118–11124. DOI: 10.1109/ICRA57147.2024.10610112.

[36] Qiao Gu et al. "ConceptGraphs: Open-Vocabulary 3D Scene Graphs for Perception and Planning". In: *2024 IEEE International Conference on Robotics and Automation (ICRA)*. 2024, pp. 5021–5028. DOI: 10.1109/ICRA57147.2024.10610243.

[37] Adolfo Guzmán. "Decomposition of a visual scene into three-dimensional bodies". In: *Proceedings of the December 9-11, 1968, Fall Joint Computer Conference, Part I*. AFIPS '68 (Fall, part I). San Francisco, California: Association for Computing Machinery, 1968, 291–304. ISBN: 9781450378994. DOI: 10.1145/1476589.1476631. URL: https://doi.org/10.1145/1476589.1476631.

[38] Ankur Handa et al. "SceneNet: An annotated model generator for indoor scene understanding". In: *2016 IEEE International Conference on Robotics and Automation (ICRA)*. 2016, pp. 5737–5743. DOI: 10.1109/ICRA.2016.7487797.

[39] Charles R. Harris et al. "Array programming with NumPy". In: *Nature* 585.7825 (Sept. 2020), pp. 357–362. DOI: 10.1038/s41586-020-2649-2. URL: https://doi.org/10.1038/s41586-020-2649-2.

[40] Kaiming He et al. "Deep Residual Learning for Image Recognition". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 770–778. DOI: 10.1109/CVPR.2016.90.

[41] Kaiming He et al. "Mask R-CNN". In: *2017 IEEE International Conference on Computer Vision (ICCV)*. 2017, pp. 2980–2988. DOI: 10.1109/ICCV.2017.322.

[42] Rui He et al. "Learning group interaction for sports video understanding from a perspective of athlete". In: *Frontiers of Computer Science* 18.4 (Dec. 18, 2023), p. 184705. ISSN: 2095-2236. DOI: 10.1007/s11704-023-2525-y. URL: https://doi.org/10.1007/s11704-023-2525-y.

[43] Greg Heinrich et al. "RADIOv2.5: Improved Baselines for Agglomerative Vision Foundation Models". In: *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2025, pp. 22487–22497. DOI: 10.1109/CVPR52734.2025.02094.

[44] Marcel Hildebrandt et al. *Scene Graph Reasoning for Visual Question Answering*. 2020. arXiv: 2007.01072 [cs.LG]. URL: https://arxiv.org/abs/2007.01072.

[45] Felix Holm et al. *CAT-SG: A Large Dynamic Scene Graph Dataset for Fine-Grained Understanding of Cataract Surgery*. 2025. arXiv: 2506.21813 [cs.CV]. URL: https://arxiv.org/abs/2506.21813.

[46] Nathan Hughes et al. "Foundations of spatial perception for robotics: Hierarchical representations and real-time systems". In: *The International Journal of Robotics Research* 43.10 (2024), pp. 1457–1505. DOI: 10.1177/02783649241229725. eprint: https://doi.org/10.1177/02783649241229725. URL: https://doi.org/10.1177/02783649241229725.

[47] Jitesh Jain et al. "OneFormer: One Transformer to Rule Universal Image Segmentation". In: *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2023, pp. 2989–2998. DOI: 10.1109/CVPR52729.2023.00292.

[48] Juanzi Li Jiaxin Shi Hanwang Zhang. "Explainable and Explicit Visual Reasoning over Scene Graphs". In: *CVPR*. 2019.

[49] Justin Johnson et al. "Image retrieval using scene graphs". In: *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2015, pp. 3668–3678. DOI: 10.1109/CVPR.2015.7298990.

[50] Andrej Karpathy and Li Fei-Fei. "Deep visual-semantic alignments for generating image descriptions". In: *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2015, pp. 3128–3137. DOI: 10.1109/CVPR.2015.7298932.

[51] Tommie Kerssies et al. "Your ViT is Secretly an Image Segmentation Model". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2025.

[52] Rawal Khirodkar et al. "Sapiens: Foundation for Human Vision Models". In: *Computer Vision – ECCV 2024: 18th European Conference, Milan, Italy, September 29–October 4, 2024, Proceedings, Part IV*. Milan, Italy: Springer-Verlag, 2024, 206–228. ISBN: 978-3-031-73234-8. DOI: 10.1007/978-3-031-73235-5_12. URL: https://doi.org/10.1007/978-3-031-73235-5_12.

[53] Daniel Kienzle et al. "COVID Detection and Severity Prediction with 3D-ConvNeXt and Custom Pretrainings". In: *Computer Vision – ECCV 2022 Workshops*. Ed. by Leonid Karlinsky, Tomer Michaeli, and Ko Nishino. Cham: Springer Nature Switzerland, 2023, pp. 500–516. ISBN: 978-3-031-25082-8.

[54] Daniel Kienzle et al. "Towards Learning Monocular 3D Object Localization from 2D Labels Using the Physical Laws of Motion". In: *2024 International Conference on 3D Vision (3DV)*. 2024, pp. 1564–1573. DOI: 10.1109/3DV62453.2024.00155.

[55] Daniel Kienzle et al. "Uplifting table tennis: a robust, real-world application for 3D trajectory and spin estimation". In: *IEEE/CVF Winter Conference on Applications of Computer Vision (WACV) 2026, 6-10 March 2026, Tucson, AZ, USA*. 2025. DOI: 10.48550/arXiv.2511.20250.

[56] Bumsoo Kim et al. "HOTR: End-to-End Human-Object Interaction Detection with Transformers". In: *CVPR*. IEEE, 2021.

[57] Nayun Kim et al. *Using 2D scene graphs as an enabler for DT topology*. 2024. DOI: 10.15480/882.13529. URL: https://hdl.handle.net/11420/49622.

[58] Nuri Kim et al. "Topological Semantic Graph Memory for Image Goal Navigation". In: *CoRL*. 2022.

[59] Alexander Kirillov et al. "Panoptic Segmentation". In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019, pp. 9396–9405. DOI: 10.1109/CVPR.2019.00963.

[60] Alexander Kirillov et al. "Segment Anything". In: *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*. 2023, pp. 3992–4003. DOI: 10.1109/ICCV51070.2023.00371.

[61] Ranjay Krishna et al. "Visual Genome: Connecting Language and Vision Using Crowdsourced Dense Image Annotations". In: *International Journal of Computer Vision* 123.1 (2017), pp. 32–73. ISSN: 1573-1405. DOI: 10.1007/s11263-016-0981-7. URL: https://doi.org/10.1007/s11263-016-0981-7 (visited on 03/06/2024).

[62] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. "ImageNet classification with deep convolutional neural networks". In: *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 1*. NIPS'12. Lake Tahoe, Nevada: Curran Associates Inc., 2012, 1097–1105.

[63] Feng Li et al. "Mask DINO: Towards a Unified Transformer-Based Framework for Object Detection and Segmentation". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2023, pp. 3041–3050.

[64] Hongsheng Li et al. "Scene Graph Generation: A comprehensive survey". In: *Neurocomputing* 566 (2024), p. 127052. ISSN: 0925-2312. DOI: https://doi.org/10.1016/j.neucom.2023.127052. URL: https://www.sciencedirect.com/science/article/pii/S092523122301175X.

[65] Li Li et al. "Domain-Wise Invariant Learning for Panoptic Scene Graph Generation". In: *ICASSP 2024 - 2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2024, pp. 3165–3169. DOI: 10.1109/ICASSP48485.2024.10447193.

[66] Li Li et al. "Panoptic Scene Graph Generation with Semantics-Prototype Learning". In: *Proceedings of the AAAI Conference on Artificial Intelligence* 38.4 (2024), pp. 3145–3153. DOI: 10.1609/aaai.v38i4.28098. URL: https://ojs.aaai.org/index.php/AAAI/article/view/28098.

[67] Li Li et al. "Panoptic Scene Graph Generation with Semantics-Prototype Learning". In: *Proceedings of the AAAI Conference on Artificial Intelligence* 38.4 (2024), pp. 3145–3153. DOI: 10.1609/aaai.v38i4.28098. URL: https://ojs.aaai.org/index.php/AAAI/article/view/28098.

[68] Rongjie Li et al. "From Pixels to Graphs: Open-Vocabulary Scene Graph Generation with Vision-Language Models". In: *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2024, pp. 28076–28086. DOI: 10.1109/CVPR52733.2024.02652.

[69] Xinye Li and Ding Chen. "A survey on deep learning-based panoptic segmentation". In: *Digital Signal Processing* 120 (2022), p. 103283. ISSN: 1051-2004. DOI: https://doi.org/10.1016/j.dsp.2021.103283. URL: https://www.sciencedirect.com/science/article/pii/S1051200421003225.

[70] Nanhao Liang et al. "CKT-RCM: Clip-Based Knowledge Transfer and Relational Context Mining for Unbiased Panoptic Scene Graph Generation". In: *ICASSP 2024 - 2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2024, pp. 3570–3574. DOI: 10.1109/ICASSP48485.2024.10446810.

[71] Tsung-Yi Lin et al. "Feature Pyramid Networks for Object Detection". In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017, pp. 936–944. DOI: 10.1109/CVPR.2017.106.

[72] Tsung-Yi Lin et al. "Microsoft COCO: Common Objects in Context". In: *Computer Vision – ECCV 2014*. Ed. by David Fleet et al. Cham: Springer International Publishing, 2014, pp. 740–755. ISBN: 978-3-319-10602-1.

[73] Xin Lin et al. "GPS-Net: Graph Property Sensing Network for Scene Graph Generation". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020.

[74] Aaron Lohner et al. "Enhancing Vision-Language Models with Scene Graphs for Traffic Accident Understanding". In: *2024 IEEE International Automated Vehicle Validation Conference (IAVVC)*. 2024, pp. 1–7. DOI: 10.1109/IAVVC63304.2024.10786395.

[75] Julian Lorenz et al. "A Fair Ranking and New Model for Panoptic Scene Graph Generation". In: *Computer Vision - ECCV 2024: 18th European Conference, Milan, Italy, September 29 - October 4, 2024, proceedings, part LXI*. Ed. by Aleš Leonardis et al. 2024, pp. 148 –164. DOI: 10.1007/978-3-031-73030-6_9.

[76] Julian Lorenz et al. "A Review and Efficient Implementation of Scene Graph Generation Metrics". In: *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. 2024, pp. 2567–2575. DOI: 10.1109/CVPRW63382.2024.00263.

[77] Julian Lorenz et al. “CoPa-SG: Dense Scene Graphs with Parametric and Proto-Relations”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) Workshops*. 2025, pp. 7604–7613.

[78] Julian Lorenz et al. *DSFlash: Comprehensive Panoptic Scene Graph Generation in Realtime*. 2026. arXiv: 2603.10538 [cs.CV]. URL: https://arxiv.org/abs/2603.10538.

[79] Julian Lorenz et al. “Haystack: A Panoptic Scene Graph Dataset to Evaluate Rare Predicate Classes”. In: *2023 IEEE/CVF International Conference on Computer Vision Workshops (ICCVW)*. 2023. DOI: 10.1109/ICCVW60793.2023.00013.

[80] Julian Lorenz et al. “On the Importance of Conditioning for Privacy-Preserving Data Augmentation”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) Workshops*. 2025, pp. 2317–2326.

[81] Ilya Loshchilov and Frank Hutter. *Decoupled Weight Decay Regularization*. 2019. arXiv: 1711.05101 [cs.LG].

[82] Cewu Lu et al. “Visual Relationship Detection with Language Priors”. In: *Computer Vision – ECCV 2016*. Ed. by Bastian Leibe et al. Cham: Springer International Publishing, 2016, pp. 852–869. ISBN: 978-3-319-46448-0.

[83] Katja Ludwig, Philipp Harzig, and Rainer Lienhart. “Detecting Arbitrary Intermediate Keypoints for Human Pose Estimation with Vision Transformers”. In: *2022 IEEE/CVF Winter Conference on Applications of Computer Vision Workshops (WACVW)*. 2022, pp. 663–671. DOI: 10.1109/WACVW54805.2022.00073.

[84] Katja Ludwig et al. “All keypoints you need: detecting arbitrary keypoints on the body of triple, high, and long jump athletes”. In: *2023 IEEE/CVF International Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), June 17 2023 to June 24 2023, Vancouver, BC, Canada*. 2023, pp. 5179 –5187. DOI: 10.1109/CVPRW59228.2023.00546.

[85] Katja Ludwig et al. “Detecting Arbitrary Keypoints on Limbs and Skis with Sparse Partly Correct Segmentation Masks”. In: *2023 IEEE/CVF Winter Conference on Applications of Computer Vision Workshops (WACVW)*. 2023, pp. 1–10. DOI: 10.1109/WACVW58289.2023.00051.

[86] Katja Ludwig et al. “Leveraging Anthropometric Measurements to Improve Human Mesh Estimation and Ensure Consistent Body Shapes”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*. 2025, pp. 5871–5880.

[87] Meta. *Aria Synthetic Environments Dataset*. Aria Synthetic Environments Dataset. https://www.projectaria.com/datasets/ase. 2024. URL: https://www.projectaria.com/datasets/ase (visited on 02/26/2025).

[88] Andrew Murray et al. *python-pillow/Pillow: 10.2.0*. Version 10.2.0. Jan. 2024. DOI: 10.5281/zenodo.10450403. URL: https://doi.org/10.5281/zenodo.10450403.

[89] Ken Museth et al. "OpenVDB: an open-source data structure and toolkit for high-resolution volumes". In: *ACM SIGGRAPH 2013 Courses*. SIGGRAPH '13. Anaheim, California: Association for Computing Machinery, 2013. ISBN: 9781450323390. DOI: 10.1145/2504435.2504454. URL: https://doi.org/10.1145/2504435.2504454.

[90] Maëlic Neau et al. "Fine-Grained is Too Coarse: A Novel Data-Centric Approach for Efficient Scene Graph Generation". In: *2023 IEEE/CVF International Conference on Computer Vision Workshops (ICCVW)*. 2023, pp. 11–20. DOI: 10.1109/ICCVW60793.2023.00008.

[91] Maëlic Neau et al. *REACT: Real-time Efficiency and Accuracy Compromise for Tradeoffs in Scene Graph Generation*. 2024. arXiv: 2405.16116 [cs.CV]. URL: https://arxiv.org/abs/2405.16116.

[92] Neo4j, Inc. *Neo4j Graph Database*. Accessed: 2025-03-05. 2025. URL: https://neo4j.com/.

[93] Alejandro Newell and Jia Deng. "Pixels to graphs by associative embedding". In: *Proceedings of the 31st International Conference on Neural Information Processing Systems*. NIPS'17. Long Beach, California, USA: Curran Associates Inc., 2017, 2168–2177. ISBN: 9781510860964.

[94] NVIDIA. *Isaac Sim*. Robotics Simulation and Synthetic Data Generation. 2025. URL: https://developer.nvidia.com/isaac-sim.

[95] Maxime Oquab et al. *DINOv2: Learning Robust Visual Features without Supervision*. Apr. 14, 2023. DOI: 10.48550/arXiv.2304.07193. arXiv: 2304.07193[cs]. URL: http://arxiv.org/abs/2304.07193 (visited on 07/20/2023).

[96] Igor Pavlov. *LZMA SDK (Software Development Kit)*. 2015. URL: https://7-zip.org/sdk.html (visited on 03/13/2024).

[97] Mrunmai Vivek Phatak et al. "HOIverse: A Synthetic Scene Graph Dataset with Human Object Interactions". In: *2025 IEEE 8th International Conference on Multimedia Information Processing and Retrieval (MIPR)*. 2025, pp. 442–448. DOI: 10.1109/MIPR67560.2025.00076.

[98] *Pickle Scanning*. URL: https://huggingface.co/docs/hub/security-pickle (visited on 06/11/2025).

[99] *pickle — Python object serialization*. Python documentation. URL: https://docs.python.org/3/library/pickle.html (visited on 06/11/2025).

[100] Antoine Pitrou. *PEP 3154 - Pickle protocol version 4 | peps.python.org*. Tech. rep. Python Enhancement Proposals, 2011. URL: https://peps.python.org/pep-3154/ (visited on 03/13/2024).

[101] Aayush Prakash et al. "Self-Supervised Real-to-Sim Scene Generation". In: *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*. 2021, pp. 16024–16034. DOI: 10.1109/ICCV48922.2021.01574.

[102] Amir M. Rahimi et al. "Toward Improving The Visual Characterization of Sport Activities With Abstracted Scene Graphs". In: *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. 2021, pp. 4495–4502. DOI: 10.1109/CVPRW53098.2021.00507.

[103] Alexander Raistrick et al. "Infinigen Indoors: Photorealistic Indoor Scenes using Procedural Generation". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2024, pp. 21783–21794.

[104] Alexander Raistrick et al. "Infinite Photorealistic Worlds Using Procedural Generation". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2023, pp. 12630–12641.

[105] Santhosh Kumar Ramakrishnan et al. "Habitat-Matterport 3D Dataset (HM3D): 1000 Large-scale 3D Environments for Embodied AI". In: *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*. Ed. by J. Vanschoren and S. Yeung. Vol. 1. 2021. URL: https://datasets-benchmarks-proceedings.neurips.cc/paper_files/paper/2021/file/34173cb38f07f89ddbebc2ac9128303f-Paper-round2.pdf.

[106] Mike Ranzinger et al. "AM-RADIO: Agglomerative Vision Foundation Model Reduce All Domains Into One". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2024, pp. 12490–12500.

[107] Shaoqing Ren et al. "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 39.6 (2017), pp. 1137–1149. DOI: 10.1109/TPAMI.2016.2577031.

[108] Lawrence G. Roberts. "Machine perception of three-dimensional solids". Accepted: 2005-08-17T19:45:17Z. Thesis. Massachusetts Institute of Technology, 1963. URL: https://dspace.mit.edu/handle/1721.1/11589 (visited on 09/02/2025).

[109] Ivan Rodin et al. "Action Scene Graphs for Long-Form Understanding of Egocentric Videos". In: *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2024, pp. 18622–18632. DOI: 10.1109/CVPR52733.2024.01762.

[110] Antoni Rosinol et al. "3D Dynamic Scene Graphs: Actionable Spatial Perception with Places, Objects, and Humans". In: *Proceedings of Robotics: Science and Systems*. Corvalis, Oregon, USA, 2020. DOI: 10.15607/RSS.2020.XVI.079.

[111] Nathan Schneider et al. "A Hierarchy with, of, and for Preposition Supersenses". In: *Proceedings of the 9th Linguistic Annotation Workshop*. Ed. by Adam Meyers, Ines Rehbein, and Heike Zinsmeister. Denver, Colorado, USA: Association for Computational Linguistics, June 2015, pp. 112–123. DOI: 10.3115/v1/W15-1612. URL: https://aclanthology.org/W15-1612/.

[112] Nathan Schneider et al. "Comprehensive Supersense Disambiguation of English Prepositions and Possessives". In: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Ed. by Iryna Gurevych and Yusuke Miyao. Melbourne, Australia: Association for Computational Linguistics, July 2018, pp. 185–196. DOI: 10.18653/v1/P18-1018. URL: https://aclanthology.org/P18-1018/.

[113] Robin Schön et al. "MMMS: Multi-Modal Multi-Surface Interactive Segmentation". In: *2025 IEEE International Conference on Content-Based Multimedia Indexing (IEEE CBMI)*. 2025. DOI: 10.48550/arXiv.2509.12963.

[114] Robin Schön et al. "Adapting the Segment Anything Model During Usage in Novel Situations". In: *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. 2024, pp. 3616–3626. DOI: 10.1109/CVPRW63382.2024.00365.

[115] Robin Schön et al. "SkipClick: Combining Quick Responses and Low-Level Features for Interactive Segmentation in Winter Sports Contexts". In: *2025 IEEE/CVF Winter Conference on Applications of Computer Vision Workshops (WACVW)*. 2025, pp. 1157–1166. DOI: 10.1109/WACVW65960.2025.00138.

[116] Xindi Shang et al. "Video Visual Relation Detection". In: *Proceedings of the 25th ACM International Conference on Multimedia*. MM '17. Mountain View, California, USA: Association for Computing Machinery, 2017, 1300–1308. ISBN: 9781450349062. DOI: 10.1145/3123266.3123380. URL: https://doi.org/10.1145/3123266.3123380.

[117] Shuai Shao et al. "Objects365: A Large-Scale, High-Quality Dataset for Object Detection". In: *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*. 2019, pp. 8429–8438. DOI: 10.1109/ICCV.2019.00852.

[118] Evan Shelhamer, Jonathan Long, and Trevor Darrell. "Fully Convolutional Networks for Semantic Segmentation". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 39.4 (2017), pp. 640–651. DOI: 10.1109/TPAMI.2016.2572683.

[119] Jongmin Shin et al. *Towards Holistic Surgical Scene Graph*. 2025. arXiv: 2507.15541 [cs.CV]. URL: https://arxiv.org/abs/2507.15541.

[120] Oriane Siméoni et al. *DINOv3*. 2025. arXiv: 2508.10104 [cs.CV]. URL: https://arxiv.org/abs/2508.10104.

[121] Kunal Pratap Singh et al. "Scene Graph Contrastive Learning for Embodied Navigation". In: *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*. 2023, pp. 10850–10860. DOI: 10.1109/ICCV51070.2023.00999.

[122] Jianlin Su et al. "RoFormer: Enhanced transformer with Rotary Position Embedding". In: *Neurocomputing* 568 (2024), p. 127063. ISSN: 0925-2312. DOI: https://doi.org/10.1016/j.neucom.2023.127063. URL: https://www.sciencedirect.com/science/article/pii/S0925231223011864.

[123] Kaihua Tang et al. "Learning to Compose Dynamic Tree Structures for Visual Contexts". In: *Conference on Computer Vision and Pattern Recognition*. 2019.

[124] Emanuel Todorov, Tom Erez, and Yuval Tassa. "MuJoCo: A physics engine for model-based control". In: *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2012, pp. 5026–5033. DOI: 10.1109/IROS.2012.6386109.

[125] Andrea Tyler and Vyvyan Evans. *The semantics of English prepositions: Spatial scenes, embodied meaning, and cognition*. Cambridge University Press, 2003.

[126] Unity Technologies. *Unity*. 2025. URL: https://unity.com/.

[127] Ashish Vaswani et al. "Attention is all you need". In: *Proceedings of the 31st International Conference on Neural Information Processing Systems*. NIPS'17. Long Beach, California, USA: Curran Associates Inc., 2017, 6000–6010. ISBN: 9781510860964.

[128] Johanna Wald et al. "Learning 3d semantic scene graphs from 3d indoor reconstructions". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 3961–3970.

[129] Johanna Wald et al. "Rio: 3d object instance re-localization in changing indoor environments". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 7658–7667.

[130] Jinghao Wang et al. "Pair Then Relation: Pair-Net for Panoptic Scene Graph Generation". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 46.12 (2024), pp. 10452–10465. DOI: 10.1109/TPAMI.2024.3442301.

[131] Weiyun Wang et al. "The All-Seeing Project: Towards Panoptic Visual Recognition and Understanding of the Open World". In: *International Conference on Representation Learning*. Ed. by B. Kim et al. Vol. 2024. 2024, pp. 24025–24057. URL: https://proceedings.iclr.cc/paper_files/paper/2024/file/68a3919db3858f548dea769f2dbba611-Paper-Conference.pdf.

[132] Tao Wu et al. "SportsHHI: A Dataset for Human-Human Interaction Detection in Sports Videos". In: *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2024, pp. 18537–18546. DOI: 10.1109/CVPR52733.2024.01754.

[133] Danfei Xu et al. "Scene Graph Generation by Iterative Message Passing". In: *Computer Vision and Pattern Recognition (CVPR)*. 2017.

[134] Shaotian Yan et al. "PCPL: Predicate-Correlation Perception Learning for Unbiased Scene Graph Generation". In: *Proceedings of the 28th ACM International Conference on Multimedia*. MM '20. New York, NY, USA: Association for Computing Machinery, 2020, pp. 265–273. ISBN: 978-1-4503-7988-5. DOI: 10.1145/3394171.3413722. URL: https://dl.acm.org/doi/10.1145/3394171.3413722 (visited on 07/19/2023).

[135] An Yang et al. "Qwen2.5 Technical Report". In: *arXiv preprint arXiv:2412.15115* (2024).

[136] Jingkang Yang et al. "4D Panoptic Scene Graph Generation". In: *Advances in Neural Information Processing Systems*. Ed. by A. Oh et al. Vol. 36. Curran Associates, Inc., 2023, pp. 69692–69705. URL: https://proceedings.neurips.cc/paper_files/paper/2023/file/dc6319dde4fb182b22fb902da9418566-Paper-Conference.pdf.

[137] Jingkang Yang et al. "Panoptic Scene Graph Generation". In: *ECCV*. 2022.

[138] Jingkang Yang et al. "Panoptic Video Scene Graph Generation". In: *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2023, pp. 18675–18685. DOI: 10.1109/CVPR52729.2023.01791.

[139] Lihe Yang et al. "Depth Anything V2". In: *Advances in Neural Information Processing Systems*. Ed. by A. Globerson et al. Vol. 37. Curran Associates, Inc., 2024, pp. 21875–21911. URL: https://proceedings.neurips.cc/paper_files/paper/2024/file/26cfdcd8fe6fd75cc53e92963a656c58-Paper-Conference.pdf.

[140] Rowan Zellers et al. "Neural Motifs: Scene Graph Parsing with Global Context". In: *Conference on Computer Vision and Pattern Recognition*. 2018.

[141] Ao Zhang et al. "Fine-Grained Scene Graph Generation with Data Transfer". In: *ECCV*. 2022.

[142] Hao Zhang et al. *DINO: DETR with Improved DeNoising Anchor Boxes for End-to-End Object Detection*. 2022. arXiv: 2203.03605 [cs.CV].

[143] Haochen Zhang et al. "VLA-3D: A dataset for 3D semantic scene understanding and navigation". In: *arXiv preprint arXiv:2411.03540* (2024).

[144] Juexiao Zhang et al. "Multiview Scene Graph". In: *The Thirty-eighth Annual Conference on Neural Information Processing Systems*. 2024. URL: https://openreview.net/forum?id=1ELFGSNBGC.

[145] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. "Open3D: A Modern Library for 3D Data Processing". In: *arXiv:1801.09847* (2018).

[146] Zijian Zhou, Miaojing Shi, and Holger Caesar. "HiLo: Exploiting High Low Frequency Relations for Unbiased Panoptic Scene Graph Generation". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. 2023, pp. 21637–21648.

[147] Zijian Zhou et al. "VLPrompt-PSG: Vision-Language Prompting for Panoptic Scene Graph Generation". In: *International Journal of Computer Vision* (Aug. 23, 2025). ISSN: 1573-1405. DOI: 10.1007/s11263-025-02564-7. URL: https://doi.org/10.1007/s11263-025-02564-7.

[148] Guangming Zhu et al. "Scene Graph Generation: A Comprehensive Survey". In: *ArXiv* abs/2201.00443 (2022).