A Commonsense Knowledge-Driven Framework for Part-of Relation Discovery in Images

Sotiris Koumourou

Thesis submitted in partial fulfillment of the requirements for the

Masters' of Science degree in Computer Science and Engineering

University of Crete
School of Sciences and Engineering
Computer Science Department
Voutes University Campus, 700 13 Heraklion, Crete, Greece

Thesis Supervisor:
Professor: *Dimitris Plexousakis*

Thesis Advisors:
Principal Researcher *Theodore Patkos*

Postdoc. Researcher *Vasilis Efthymiou*

This work has been performed at the University of Crete, School of Sciences and Engineering, Computer Science Department.

The work has been supported by the Foundation for Research and Technology - Hellas (FORTH), Institute of Computer Science (ICS).

UNIVERSITY OF CRETE
COMPUTER SCIENCE DEPARTMENT

A Commonsense Knowledge-Driven Framework for Part-of Relation Discovery in Images

Thesis submitted by
Sotiris Koumourou
in partial fulfillment of the requirements for the
Masters' of Science degree in Computer Science

THESIS APPROVAL

Author: ______________________
Sotiris Koumourou

Committee approvals: ______________________
Dimitiris Plexousakis
Professor, Thesis Supervisor

____
Antonios Argyros
Professor, Committee Member

____
Theodore Patkos
Principal Researcher, Committee Member

Departmental approval: ______________________
Polyvios Pratikakis
Associate Professor, Director of Graduate Studies

Heraklion, November 2023

A Commonsense Knowledge-Driven Framework for Part-of Relation Discovery in Images

Abstract

The integration of data-driven techniques with knowledge-based methods has led to many accomplishments in the past few years. While machine learning techniques have proven their potential in diverse domains, they often display weaknesses that symbolic methods can help overcome.

This thesis focuses on the development of a framework tailored for the detection of $partOf$ relations in images through the integration of commonsense knowledge. A $partOf$ relation represents the association between an entity and its component or constituent part. The primary objective of this framework is to enhance the precision of relation predictions by incorporating external information derived from problem-agnostic knowledge graphs, specifically ConceptNet. The proposed framework, named *PReDeCK*, relies on graphs for capturing structured, semantic knowledge about commonsense notions. By conducting symbolic reasoning over the extracted information utilizing the expressive capabilities offered by Answer Set Programming (ASP), *PReDeCK* can eliminate counter-intuitive conclusions and improve the accuracy of relation predictions.

Additionally, this thesis addresses the challenge of identifying errors in the outputs generated by object detection models. We expand *PReDeCK* to detect potential errors by cross-referencing the model's results with established real-world knowledge. Moreover, techniques for addressing identified errors are also outlined.

We perform experimental evaluations to assess the performance of the framework by using a well-known image dataset, Semantic Pascal-Part, which includes a diverse range of everyday objects and their constituent parts. Throughout the experimental phase, we analyze various framework variations to highlight the critical role of a robust and accurate knowledge domain, as well as the impact of incomplete or erroneous data on the results. The outcomes of these experiments verify that the synergy between visual data and commonsense knowledge leads to a significant improvement in the precision of relation detection. Additionally, the results regarding the error detection task, are quite promising, marking an initial step in quality assurance for object detection models.

Overall, this thesis provides insights into a commonsense knowledge-driven framework for discovering $partOf$ relations in images and for identifying potential errors in the outputs generated by an object detection model through the use of symbolic reasoning.

Μεθοδολογία για τον εντοπισμό σχέσεων μέρους-όλου σε εικόνες αξιοποιώντας γνώση κοινής λογικής

Περίληψη

Ο συνδυασμός των τεχνικών βασισμένων στα δεδομένα με μεθόδους βασισμένες στη γνώση, έχει οδηγήσει σε σημαντικά επιτεύγματα τα τελευταία χρόνια. Ενώ οι τεχνικές μηχανικής μάθησης έχουν αποδείξει τη δυναμική τους σε διάφορους τομείς, συχνά εμφανίζουν αδυναμίες που μπορούν να αντιμετωπιστούν με τις συμβολικές μεθόδους.

Η παρουσιαζόμενη εργασία επικεντρώνεται στην ανάπτυξη μιας μεθοδολογίας, η οποία είναι προσαρμοσμένη για τον εντοπισμό σχέσεων μέρους-όλου σε εικόνες μέσω της ένταξης γνώσης κοινής λογικής. Ο κύριος στόχος αυτής της μεθοδολογίας είναι η βελτίωση ακρίβειας στον εντοπισμό σχέσεων με την ενσωμάτωση εξωτερικής πληροφορίας που προέρχεται από γράφους γνώσης που είναι ανεξάρτητοι από το πρόβλημα. Η προτεινόμενη μεθοδολογία, με την ονομασία ***PReDeCK***, βασίζεται σε ανοικτούς γράφους δεδομένων για την αποτύπωση της γνώσης κοινής λογικής, και συγκεκριμένα από τον ανοικτό γράφο ConceptNet. Μέσω συμβολικών μεθόδων συμβολιστικής που αξιοποιούν την εκφραστικότητα της Answer Set Programming (ASP), το ***PReDeCK*** χρησιμοποιεί αυτή τη πληροφορία για να εξαλείψει αντίθετα συμπεράσματα και να βελτιώσει την ακρίβεια σχέσεων που εντοπίζει.

Επιπλέον, η παρούσα εργασία αντιμετωπίζει την πρόκληση του εντοπισμού σφαλμάτων στα αποτελέσματα που παράγονται από μοντέλα ανίχνευσης αντικειμένων. Επεκτείνουμε το ***PReDeCK*** για τον εντοπισμό πιθανών σφαλμάτων διασταυρώνοντας τα αποτελέσματα του μοντέλου με εδραιωμένες γνώσεις του πραγματικού κόσμου. Επιπλέον, παρουσιάζονται τεχνικές για την αντιμετώπιση των εντοπισμένων σφαλμάτων.

Πραγματοποιούμε πειραματικές διαδικασίες για την αξιολόγηση της απόδοσης της μεθοδολογίας μας, χρησιμοποιώντας ένα ευρέως γνωστό σύνολο εικόνων, το Semantic Pascal-Part , το οποίο περιλαμβάνει μια ποικιλία καθημερινών αντικειμένων και τα αντίστοιχα μέρη τους. Κατά τη διάρκεια της πειραματικής φάσης, αναλύουμε διάφορες παραλλαγές της μεθοδολογίας, για να υπογραμμίσουμε τον κρίσιμο ρόλο ενός ορθού και ακριβούς αποθετηρίου γνώσης, καθώς και τον αντίκτυπο των ατελών ή εσφαλμένων δεδομένων στα αποτελέσματα.

Τα αποτελέσματα αυτών των πειραμάτων επιβεβαιώνουν ότι ο συνδυασμός οπτικών δεδομένων και γνώσης κοινής λογικής, οδηγεί σε σημαντική βελτίωση στην ακρίβεια της ανίχνευσης των σχέσεων. Επιπλέον, τα αποτελέσματα σχετικά με την ανίχνευση πιθανών σφαλμάτων είναι αρκετά ικανοποιητικά, παρέχοντας ένα αρχικό βήμα στην διασφάλιση της ποιότητας για τα μοντέλα ανίχνευσης αντικειμένων.

Συνολικά, η εργασία παρουσιάζει πώς υλοποιείται μια μεθοδολογία που αξιοποιεί γνώση κοινης λογικής από ανοιχτούς γράφους δεδομένων, για να ανιχνεύσει σχέσεις μέρους-όλου μεταξύ αντικειμένων που ανιχνεύονται στις εικόνες, και για να εντοπίσει πιθανά σφάλματα στα αποτελέσματα που παράγονται από ένα μοντέλο ανίχνευσης αντικειμένων, μέσω της χρήσης του συμβολικού συλλογισμού.

Ευχαριστίες

Στο σημείο αυτό αισθάνομαι την ανάγκη να εκφράσω τις ειλικρινείς και θερμές ευχαριστίες μου σε όσους συνέβαλλαν στην ολοκλήρωση αυτής της προσπάθειας. Πρώτα απ' όλα θέλω να ευχαριστήσω τους επιβλέποντές καθηγητές μου, κύριο Θοδωρή Πάτκο και κύριο Βασίλη Ευθυμίου για την εξαιρετική καθοδήγηση και τις συμβουλές τους καθ΄ όλη τη διάρκεία της εκπόνησης αυτής της διπλωματικής. Επίσης, ευχαριστώ τον επόπτη καθηγητή μου, κύριο Δημήτρη Πλεξουσάκη για την πολύτιμη συμβολή του. Τέλος, θέλω να ευχαριστήσω τους γονείς μου, όλη την οικογένεια μου, καθώς και τους φίλους μου που είναι πάντα στο πλευρό μου σε κάθε βήμα της ζωής μου.

Contents

Table of Contents **i**

List of Tables **v**

List of Figures **ix**

1 Introduction **1**

2 Background **5**
- 2.1 Knowledge Graphs 5
- 2.2 Answer Set Programming (ASP) 6
- 2.3 NeurASP 6
- 2.4 Pascal-Part and Semantic Pascal-Part Datasets 7
- 2.5 Terms and Notation 7

3 Related Work **9**
- 3.1 Neurosymbolic AI 9
- 3.2 Visual Relationships Detection 9
- 3.3 Enhancing Visual Tasks with Knowledge Graphs 10

4 Detecting $partOf$ Relations **13**
- 4.1 Problem Description 13
- 4.2 Overview of the Methodology 15
- 4.3 Components Analysis 15
  - 4.3.1 Knowledge Filtering Module 15
  - 4.3.2 Object Detection Model 18
  - 4.3.3 Reasoning Module 19
- 4.4 Example 23
- 4.5 Summary 24

5 Identifying Errors in Object Detector's Output **25**
- 5.1 Problem Description 25
- 5.2 Overview of the Methodology 26
- 5.3 Error Detection and Handling 26

5.3.1 Error Case P 27
5.3.1.1 Error Detection Methodology 28
5.3.1.2 Error Handling Methodology 28
5.3.2 Error Case O 29
5.3.2.1 Error Detection Methodology 30
5.3.2.2 Error Handling Methodology 31
5.3.3 Error Case PO 31
5.3.3.1 Error Detection Methodology 32
5.3.3.2 Error Handling Methodology 32
5.3.4 Error Case MO 33
5.3.4.1 Error Detection Methodology 34
5.3.4.2 Error Handling Methodology 35
5.3.5 Wrong detection coordinates 38
5.3.6 Interesting Cases 39
5.4 Summary 42

6 Implementation 45
6.1 Framework set-up 46

7 Evaluation 49
7.1 Setting 49
7.2 Design of Experiments 49
7.2.1 Evaluation Measures 49
7.2.2 Hypotheses 51
7.3 Experimental Results 53
7.3.1 Object Detection Model 54
7.3.2 *PartOf* Relations Discovery 54
7.3.2.1 *PReDeCK-Visual Baseline* (Hypothesis 1) 54
7.3.2.2 *PReDeCK-PReDeCK*$_{GT}$ (Hypothesis 2) 55
7.3.2.3 *PReDeCK - PReDeCK*$_{CN\ variations}$ (Hypothesis 3) 55
7.3.2.4 Alternating IoU Value 56
7.3.3 Error Detection 58
7.3.3.1 *PReDeCK+ED*$_{CN}$ - *PReDeCK+ED*$_{GT}$ (Hypothesis 1) . 58
7.3.3.2 Deploying various object detection models (Hypothesis 2) 58
7.3.3.3 Error Handling 60

8 Discussion 61
8.0.1 Limitations 62
8.0.2 Pre-trained Models Association 63

9 Conclusions 65

Bibliography 67

Appendices **71**

A ASP Auxiliary Rules **73**
- A.1 Bounding Box Overlap Ruleset: 73
- A.2 Knowledge ($meronym$) Enrichment Ruleset 74
- A.3 *PartOf* Detection Ruleset 75
- A.4 Error Detection Ruleset 76
  - A.4.1 Error MO Handling Ruleset 77

B Cypher Queries **81**

C Part Of ($meronym$) relations in ground truth **83**

List of Tables

2.1 Table of Terms. 8

5.1 Error Cases. 27

7.1 Object Detection Models Performance. 54
7.2 Total True positives (TP), False positives (FP) and False negatives (FN). . 57
7.3 Precision (P), recall (R) and F1 with IoU = 0.5. 57
7.4 Precision (P), recall (R) and F1 with IoU = 0.3. 57
7.5 Error Case P results. 59
7.6 Error Case O results. 59
7.7 Error Case PO results. 59
7.8 Error Case MO results (MO1). 59
7.9 Error Case MO additional evaluation methods Precision (P). 60

List of Figures

1.1 A pair of images, where whole-part relations among bounding boxes need to be identified. 4

4.1 Component diagram of our proposed neuro-symbolic approach to $partOf$ relation detection in images. 16

4.2 Subgraph with the extracted information from ConceptNet regarding the label “Bird”. In the subgraph, a number of relations connected with “Bird” are shown. For instance, the triples (wing,partOf,bird), (beak,partOf,bird), (bird,isA,animal), etc., exist in the subgraph. 18

4.3 Example of an object detector’s input (left) and output (right). 19

5.1 Error detection and Handling methodology outline. 27

5.2 Error Case P example. In the image, an object part classified as “Handlebar” is detected with low confidence, that can be spatially connected with the detected whole object “Person”. Because the two cannot be semantically associated, the framework identifies a case P error triggered by the “Handlebar” bounding box. 29

5.3 Error Case P handling example. The label “Handlebar” of the bounding box that triggered the error is correctly changed to “Hand”. 30

5.4 Error Case O example. The error is triggered by the whole object “Aeroplane”. This happens because “Aeroplane” is detected with low confidence and is spatially associated with the bounding boxes labeled as “Animal wing”, which have been detected with high confidence. 31

5.5 Error Case O handling example. To resolve the error, the framework changes the label “Aeroplane” to “Bird”. 32

5.6 Error Case PO example. The detection of spatially connected “Person” and “Muzzle” objects, both with low confidence, led to the event of the case PO error. 33

5.7 Error Case PO handling example. The framework addresses the error by changing the “Muzzle” label to “Head”. Following the label modification, there is no violation within the knowledge domain. Consequently, the “Person” box, while visually incorrect, remains unalterable. 34

5.8 Error Case MO example 1. Multiple high-confidence object parts are detected (e.g., “Nose”,“Muzzle”, etc.) that cannot be spatially connected with any whole object within the image. 35
5.9 Error Case MO example 2. The object parts “Wheel” and “Hand” are detected with high confidence, but cannot be related with a whole object. . 36
5.10 Error Case MO handling example 1. The framework reports that at least 1 whole object with label “Person” should have been returned by the object detector. 37
5.11 Error Case MO handling example 2. The framework reports that at least 2 whole objects are missing. The possible classes for one of the undetected whole objects include “Aeroplane”, “Bicycle”, etc. The only possible label for the other missing object is “Person”. 37
5.12 Example image where errors are detected due to inaccurate bounding boxes coordinates. The bounding box of the “Person”, should be larger to include the object parts “Hand” and “Foot”. 38
5.13 Errors Filtering Pipeline. 39
5.14 A low-confidence (yellow) bounding box “Nose” is detected that is spatially associated with a detected whole object “Sheep”. An error case P is triggered, because in the ground truth there is no $meronym$(nose,sheep), hence the yellow box was misclassified. However, a sheep having a nose is not against commonsense knowledge. Using the knowledge domain constructed via ConceptNet, this error cannot be detected. 40
5.15 An object part “Body” (yellow box) is detected by the object detection model and causes a case P error. However, no object exists in the ground truth annotations regarding the yellow bounding box. Hence, this error detection case was incorrect. 41
5.16 Both a “Bicycle” and a “Leg” are detected with confidence scores below the threshold in the image (left), leading to the correct identification of an error case PO due to their lack of semantic connection. However, when the framework attempts to rectify the error (right), the fact that “Leg” is assigned a higher confidence score than “Bicycle” results in the relabeling of the “Bicycle” as “Person”. This handling outcome can be described as “neutral”, as the “Bicycle” is wrongly dropped, but a “Person” is correctly classified. 42
5.17 Figure showcases a scenario where our error-handling process results in an incorrect label change. Specifically, a “Wheel” object part (indicated by the yellow box) is detected with a confidence score below the specified threshold in the left image. This “Wheel” is spatially connected to a detected “Person”, though no semantic validation exists. The object detection model fails to recognize the existing “Bicycle” in the image, leading our methodology to trigger an erroneous error case P. In an attempt to rectify this error, the handling process inaccurately relabels the “Wheel” as “Chair”, as depicted in the right image, ultimately decreasing the overall object detection performance. 43

5.18 Object detector identifies a “Headlight” (red box) with high confidence, but does not relate it to a whole object, triggering an MO error case. The missing object is likely a vehicle (e.g., car), but such object is not annotated in the dataset. Examining such cases may reveal incorrect or missing annotations in the dataset. 44

6.1 *PReDeCK*’s Execution Pipeline 46

Chapter 1

Introduction

The rise of Machine Learning (ML) has been nothing short of revolutionary. In the past decade, ML has seen tremendous growth thanks to the increase in computing power and the vast data availability. Neural networks have emerged as one of the most promising techniques in the field of ML due to their ability to learn from data and generalize patterns. Neural networks' abilities have led to groundbreaking achievements in various fields, such as natural language processing, computer vision, and many more.

However, despite the remarkable success, the limitations of ML approaches are well-recognized, and the need to couple these techniques with approaches that exploit structured knowledge and symbolic reasoning, is increasingly attracting interest [17, 20, 16].

In the field of Computer Vision in particular, the combination of low-level perception, high-level reasoning, and commonsense knowledge exploitation is a highly desirable objective [48]. Prominent results come from approaches that aim to combine advancements in neural networks with expressive, formal logics; yet, most approaches either focus on rather restricted domains, such as sudoku puzzles where the domain rules are explicit and can easily be written in a formal language (e.g., [42, 31]), or utilize rules custom-made by experts (e.g., [2, 9]). However, these domain-specific approaches raise questions about their ability to generalize beyond their particular domains.

In this thesis, we introduce a neuro-symbolic approach for a non-trivial vision task, that of identifying $partOf$ relations among objects detected in an image. Our methodology avoids overly simplistic assumptions and, instead, harnesses commonsense knowledge from open repositories. Following the direction adopted by recent studies, as in [42], we separate perception and reasoning. Initially, we employ a standard neural model for simple perception tasks like object detection. Then, we rely on high-level, expressive rules for more intricate domain inference tasks, which involve complex reasoning and contextual understanding. This separation allows us to effectively address both basic and complex aspects of a problem domain. We thus avoid learning complex domain rules and constraints with an end-to-end model, when these rules can be extracted from commonsense repositories, leading to significant simplifications of the learning process. This way, we also eliminate the need for the laborious task of generating extensive training data. We therefore utilize more robust perception, while working towards the "small data for big tasks" paradigm [48].

To accomplish this objective, relying on pre-designed domain rules does not suffice, as it may give the impression that expert knowledge engineering is tailored for the specific problem. Therefore, a secondary aim of this thesis is to investigate how easily and, more important, efficiently one can extract commonsense domain knowledge that can be proven fruitful in enhancing scene understanding. Specifically, we introduce a methodology that harnesses an open, problem-agnostic knowledge graph, ConceptNet [36], to construct the knowledge domain through data retrieval, cleansing, and logical inference techniques.

Figure 1.1 (top) shows a simple example regarding the target problem, where the "Hand" bounding boxes, which are the rectangular areas enclosing the hands, should be characterized as having a whole-part relation with the bounding box of the "Person", and not with the "Bicycle" bounding box, despite the overlap; the situation gets more involved in information-rich environments, as shown in Figure 1.1 (bottom) (the labels of the bounding boxes are omitted). It should be noted that our objective is not to compare our method with a dedicated classifier trained extensively on a dataset annotated with $partOf$ relations, rather to use a typical, out-of-the-box object classifier, and address the problem of $partOf$ identification between objects without any training data, but exclusively relying on logical reasoning over problem-agnostic commonsense knowledge. After all, despite the focus on the $partOf$ relation, solutions to the given problem can be used to represent a large class of relations, as mentioned in [18].

Additionally, we expand our approach to leverage its outputs for identifying potential errors in the object detection model, guided by a commonsense perspective. By utilizing the earlier mentioned inferences, we can identify various instances where the object detection model produces inaccuracies. Our framework not only detects errors but also strives to categorize and address them based on their types, potentially enhancing the overall performance of the object detection model.

Our solution deploys a popular off-the-shelf model for object detection, which we fine-tuned on a generic dataset (Semantic Pascal-Part [8]), and we couple it with commonsense knowledge obtained by our proposed method, in order to derive the $partOf$ relations. We use the NeurASP [42] framework to combine neural networks with logical reasoning; yet, our methodology is generic enough to be used with other repositories and reasoning tools.

The main contributions of this work are the following:

- We develop a hybrid approach that relies on state-of-the-art neuro-symbolic tools, which enables the complement of data-driven methods with external knowledge obtained from commonsense repositories, as well as with logical inference. Applied to the challenge of detecting $partOf$ relations among identified objects in an image, even in the absence of specific training data for these relations, we systematically assess its performance across various perspectives.

- We present a generic methodology for extracting knowledge from commonsense knowledge graphs, which exploits both topological aspects and semantic relations, and study how well we can reduce noise in the data, while keeping high recall score.

- An error detection methodology is showcased that cross-references the model's detections with real-world knowledge, effectively flagging instances of errors for further examination. Our proposed framework reports high performance scores tackling the error detection task, even when object detection models were trained on noisy data, providing a first step in quality check of an object detection model.
- In our experimental assessments, we demonstrate that integrating visual data with commonsense knowledge, considerably outperform the results of the baselines and is comparable to methods relying on manually annotated datasets. This enhancement results in improved precision with minimal impact on recall.

All data and source code referred to in this work are publicly available.[1]

Outline. The rest of this thesis is structured as follows. Chapter 2 provides some background knowledge that is necessary for this thesis, while Chapter 3 reviews related works. In Chapters 4 and 5, we provide the details of our methodologies to detect $partOf$ relations between objects detected in an image, and identify possible errors of an object detector, respectively. Chapter 6 covers the details of our implementation. The evaluation process of our work and our findings are presented in Chapter 7. Finally, in Chapter 9, we conclude this thesis by summarizing the key outcomes of our research.

[1]`https://github.com/Kmrs97/PReDeCK`

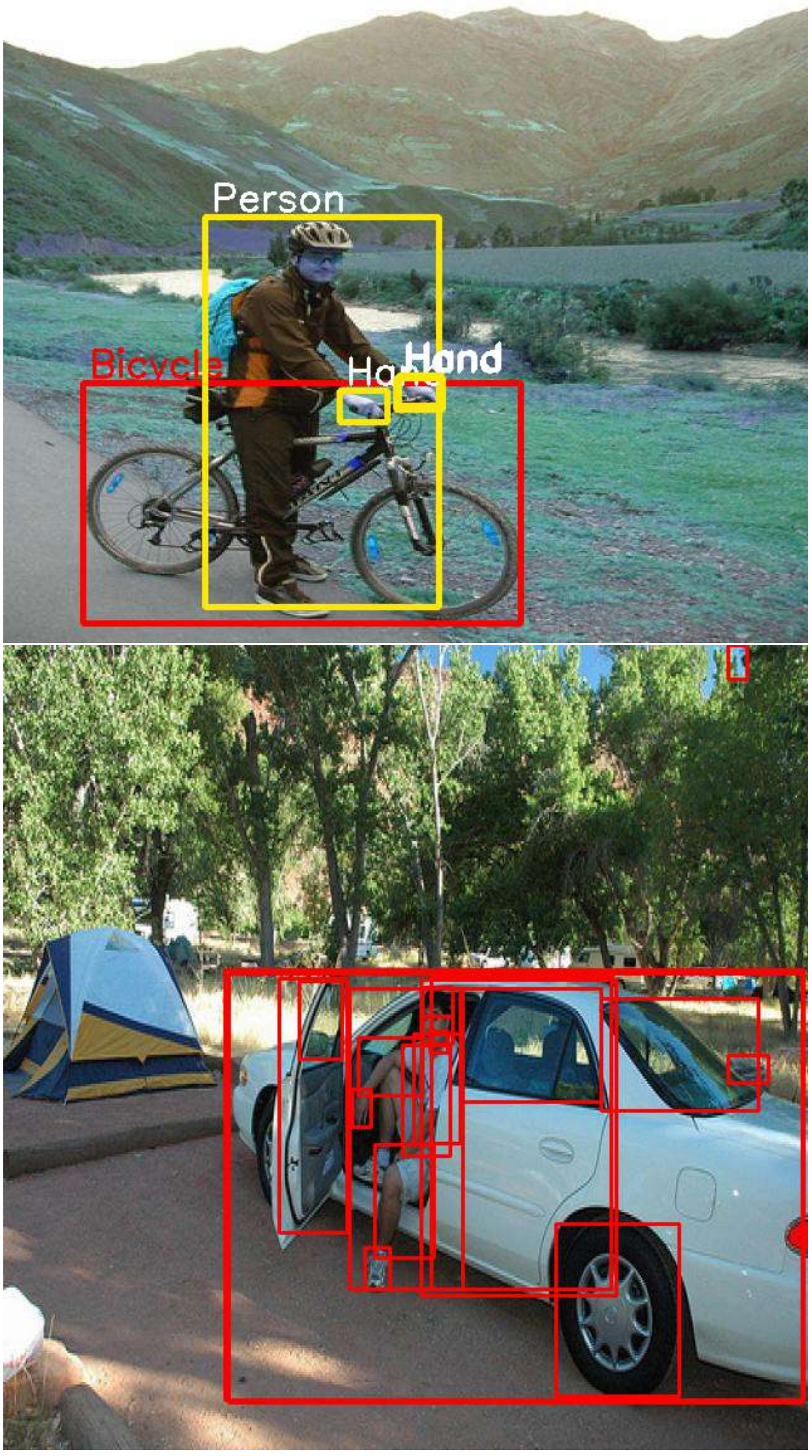


Figure 1.1: A pair of images, where whole-part relations among bounding boxes need to be identified.

Chapter 2

Background

This chapter provides an in-depth exploration of the fundamental principles and tools employed in this work. After introducing some basic concepts and tools, at the end of this chapter we provide a table of terms and notation used throughout this work.

2.1 Knowledge Graphs

Traditional approaches for detecting $partOf$ relations rely entirely on visual features identified in the images, but require a wealth of annotated data to train their models. In this work, we study how commonsense knowledge and inferencing can be exploited to improve the accuracy of the results for the aforementioned problem.

Let's consider again the example of Figure 1.1. Note that both hands are fully enclosed by the bounding boxes of both the person (big yellow rectangle) and the bicycle (red rectangle). A popular approach is to consider only the inclusion proportion of the bounding boxes, which in this case will erroneously return that the hands may be part of the bicycle. By incorporating commonsense knowledge, i.e., the knowledge that among the two candidate objects only a person can have hands, we can assist a system in eliminating certain counter-intuitive conclusions.

For capturing commonsense knowledge, we rely on knowledge graphs. A *knowledge graph* (KG) can be characterized as a set of (subject, predicate, object) triples that form a named graph. Subject and object are nodes of the graph, while every triple defines an edge between the corresponding nodes, with the label of that edge being the predicate name of that triple.

ConceptNet [36] is an open KG that captures a wide range of commonsense knowledge about the world, including common relations (e.g., "PartOf", "IsA"), between concepts. This KG has been used for a variety of AI tasks, such as word sense disambiguation [35], question answering [38] and more [34, 24]. A big portion of ConceptNet has been populated manually, which unavoidably has introduced noise in the data.

2.2 Answer Set Programming (ASP)

Answer Set Programming (ASP) [29] is a declarative programming paradigm that allows the user to solve complex computational problems, such as planning, optimization, and knowledge representation. To solve these problems, a knowledge engineer models the domain with a set of rules and constraints; then, a reasoner generates zero or more *answer sets*, which are valid solutions to the problem.

Example ASP rules are the following:

$$n(a). \tag{2.1}$$

$$p(X) :- n(X). \tag{2.2}$$

$$:- not \;\; n(a). \tag{2.3}$$

$$:- not \;\; n(\_). \tag{2.4}$$

Rule 2.1 is a *fact* stating that constant a is an instance of predicate n, i.e., a has the property n. Rule 2.2 in an inference rule, denoting that if variable X is instantiated with a constant that has the property n, then this instance of X also has the property p.[1] Rule 2.3 is a *constraint* rule stating that no possible solution should exist that contains the predicate n with a as its argument. In Rule 2.4, one can see the use of the underscore "_", which in ASP serves as a versatile placeholder for unspecified or unknown values. Hence, in this case, the constraint rule states that no possible solution should exist that contains the predicate n no matter what its argument is.

2.3 NeurASP

One of the most efficient ASP reasoning systems is *Clingo* [15], which is used for the grounding and solving of logic programs. NeurASP [42], an extension of Clingo, is a novel framework that combines the strengths of ASP and Neural Networks (NNs), enabling the development of hybrid models that can reason with high-level knowledge, while adapting to data-driven learning. NeurASP leverages the two paradigms to address complex tasks that require both data-driven prediction and reasoning.

It consists of two interconnected components: the ASP module and the NN module. The ASP module enables the encoding of logical rules and constraints, formulated in ASP, that define the domain knowledge and reasoning tasks. These rules are grounded into logical atoms. The NN module comprises customizable neural networks that take these logical atoms as input and perform computations (e.g., classification, regression) to generate desired outputs. A neural network is represented as an ASP atom in NeurASP, whose arguments capture the possible outcomes of the NN.

For our purposes, we use NeurASP for inferencing, i.e., to make derivations based on the estimations of a NN, but the framework can also be used for training the NN using logical rules in combination with numerical data.

[1]Predicates and constants start with a lowercase character, while variables inside predicates start with an uppercase character. All variables are implicitly universally quantified in formulae.

2.4 Pascal-Part and Semantic Pascal-Part Datasets

The Pascal-Part Dataset [5] is a widely used dataset in computer vision that focuses on object segmentation and part localization. It was used to develop advanced computer vision algorithms for various applications, such as image understanding [4], object detection [47] and scene understanding [43].

Pascal-Part is an extension of the Pascal VOC dataset [11], which provides additional annotations that label the parts of the objects within the images. The dataset consists of a variety of diverse images that cover 20 object categories, such as “dog”, “person” and “car”. The parts for each object category are also annotated in the dataset. Examples of the part labels are “head”, “torso”, “headlight”, and “wheel”, depending on the object category.

In our work, we utilize the Semantic Pascal-Part Dataset [8], which is the RDF version of the Pascal-Part Dataset. In the latter, the labels for the individual parts are too specific for many applications, e.g., “left lower arm”. The Semantic Pascal-Part Dataset generalizes these annotations by merging the segments of the images that refer to the same part into a unique segment, e.g., two segments labeled with “left lower arm” and “left front arm” of the same arm are merged into a segment labeled as “arm”. Finally, these segments are converted into bounding boxes. The Semantic Pascal-Part Dataset also offers the Pascal-Part OWL ontology that formalizes the *part of* relationships between the objects and their parts. The whole set of the $partOf$ relations present in the ontology is presented in Appendix C.

2.5 Terms and Notation

In Table 2.1, one can find important terms that are used throughout this thesis, along with their informal explanations.

Our approach is directly reliant on the object detection task, performed by neural networks. *Object detection* is the task of identifying objects present in a given image. Every *object* in an image is described as a tuple (l, bb), where l is its *label* and bb is a *bounding box*, i.e., a rectangular area defined by the top-left and the bottom-right coordinates of the detected object in the input image.

In our research, every label belongs to one of the following two types: “Object” or “Part”. The classification of labels is provided by the input dataset. Based on the type of their label, we divide objects into *whole objects* (objects whose label is of type “Object”) and *(object) parts* (objects whose label is of type “Part”). Intuitively, whole objects denote detected objects that may consist of other objects, while object parts refer to detected objects that may constitute a component of a whole object. To prevent erroneous associations between objects from differing contexts, we utilize ConceptNet’s *label category*, which assigns context-specific sense labels to its concepts.

The framework proposed in this thesis is named *PReDeCK*, which stands for “Part of Relations Detection using Commonsense Knowledge”. The primary objective of *PReDeCK* is to identify $partOf$ relations between detected objects within an image. The

Table 2.1: Table of Terms.

Term	Explanation
bounding box (bb)	a rectangular area within an image, defined by its top-left and bottom-right coordinates (width and height), i.e., four integers
label (l)	the classification of the contents of a bounding box (e.g., person, hand, bicycle)
object	a bounding box bb associated with a label l, i.e., an (l, bb) tuple
label type	labels are divided into whole-object and object-part labels ("Objects" and "Parts")
whole object	an object whose label is of type whole-object (e.g., a person)
(object) part	an object whose label is of type object-part (e.g., a hand)
label category	the categorization label that exists in ConceptNet, e.g., Cat is of category Animal
$partOf$	a relation between an object part p and a whole object o in a given image i, denoting that p is a component of o in i; these are the relations that we aim to infer
$hasObject$	a relation between an image and some object within that image, returned by an object detector
$meronym$	a meronym relation between two labels, e.g., $meronym$(hand, person)
PReDeCK	our methodology that detects $partOf$ relations
PReDeCK+ED	an expansion of PReDeCK that detects errors in $hasObject$ relations

$partOf$ relation defines that an object part p is part of a whole object o existing in an image. To semantically verify possible $partOf$ relations, our framework utilizes the $meronym$ relations (e.g., $meronym$(mouth, person)) within the knowledge domain constructed using our methodology, which comprises a collection of ASP-formulated triples.

Additionally, *PReDeCK+ED* represents an extension of our framework, designed to identify potential errors in the outputs generated by the object detection model.

Chapter 3

Related Work

In the following section of this thesis a comprehensive overview of prior research and literature that is relevant to the topic under investigation is provided. This review allows us to delve into the existing body of knowledge, offering valuable context for our study, identifying gaps and areas for further exploration, and establishing the foundation upon which our own contributions and findings are built.

3.1 Neurosymbolic AI

Neurosymbolic AI is the field of research and applications that combines machine learning methods, based on artificial neural networks, with symbolic approaches to computing and Artificial Intelligence (AI) [20]. This multidisciplinary approach aims to bridge the gap between the two paradigms to create intelligent systems that integrate the advantages of both worlds [16]. In the past years, the significance of neurosymbolic AI has grown substantially. The domain has gathered increasing attention from the research and academic communities. Researchers all over the world recognise the potential of Neurosymbolic AI to address some of the most challenging open problems in AI. This notable interest has led to various applications of the Neurosymbolic framework that tackle many real-life tasks in numerous domains [27, 3].

3.2 Visual Relationships Detection

Visual Relationships Detection (VRD) is a cutting-edge field in computer vision and artificial intelligence that focuses on understanding the complex interactions and connections between objects within images or videos. This research area seeks to uncover the intricate relationships that exist among various elements present in visual data, such as identifying actions, interactions, or spatial arrangements between objects. The applications of VRD are wide-ranging with transformative potential across various domains [19, 25, 39]. In [41], for example, they propose a hierarchical graph convolutional network along with a graph attention method for Visual Dialogue Generation. In the context of Image Captioning, Yao et al. [44] introduce a combination of Graph Convolutional Networks and the Long Short-Term Memory architecture, seamlessly integrating semantic and spatial object relationships into the image encoder.

3.3 Enhancing Visual Tasks with Knowledge Graphs

Systems that address Visual Tasks, often show weaknesses due to their inability to perform complex reasoning and understand complex relationships within visual data. Some key weaknesses are their lack of contextual understanding, ambiguity handling, limited generalization, incomplete information, and more. Importing knowledge from knowledge graphs addresses these weaknesses by providing systems with structured, external knowledge. It empowers visual systems to perform more sophisticated reasoning, better understand contextual information, handle ambiguity, and generalize knowledge. This integration enhances the overall capabilities of these systems, making them more versatile and capable of addressing a wider range of visual tasks with improved accuracy and depth.

In [12], the authors introduce a framework that combines knowledge graphs with object detection algorithms. Focusing on the key computer vision task of image-based object detection, the system extracts and generalizes knowledge from these graphs. It utilizes semantic consistency metrics from the knowledge graph to enhance detection, ensuring improved recall and consistent background knowledge without sacrificing mean average precision. Díaz-Rodríguez et al. [7] present the X-NeSyL methodology, which combines deep learning representations with expert knowledge graphs to achieve explainable neural-symbolic learning. The system utilizes knowledge graphs to represent symbolic expert knowledge, which is then leveraged by a deep learning model. The use of knowledge graphs enhances interpretability and allows for the alignment of symbolic and neural representation learning components. The methodology is demonstrating improved performance and explainability compared to traditional training methods.

Commonsense Knowledge. Many approaches use knowledge repositories to integrate commonsense knowledge to systems because it further enhances their ability to reason, make better predictions, and exhibit more human-like behavior. NeuSyRE [26] is a neuro-symbolic visual understanding and reasoning framework that enhances scene graphs for improved downstream reasoning in visual scenes. The system employs a deep neural network-based pipeline for object detection and relationship prediction to generate scene graphs. It leverages heterogeneous knowledge graphs, such as ConceptNet and WordNet, to enrich the scene graphs with commonsense knowledge, providing background information and related facts about visual concepts. The integration of knowledge graphs and commonsense knowledge enhances the expressiveness and autonomy of visual understanding and reasoning, leading to improved performance in tasks such as image captioning and image generation.

For the detection of $partOf$ relations in particular, the work in [9] utilizes Logic Tensor Networks (LTN) based on the Real Logic formalism, in order to perform classification and $meronym$ relations discovery between the detected objects. That system was also evaluated on the Pascal-Part Dataset, as is ours. While the rationale is similar to ours, in this work the commonsense knowledge was derived from WordNet and infused in the system manually. In contrast, our suggested approach automatically incorporates the information, and is more generic making it suitable to be applied to a variety of KGs and domains with minimum effort. What's more, the use of a non-monotonic language, such as ASP, offers higher expressivity in modeling commonsense domains (e.g., defaults) in comparison to first-order logics, such as Real Logic. It should be noted though that tools, such as LTNs and NeurASP, present a prominent prospect in addressing tasks that require neuro-symbolic approaches and state-of-the-art research in this direction is expected to produce even better results in the future.

Identifying errors of an object detection model on images, may be proven very useful and offer various insights. CSK-SNIFFER [14] is a system showcasing the application of commonsense knowledge for the automated prediction of failures in object detection models when analyzing extensive image datasets. This system promotes human-AI collaboration by pinpointing errors through spatial commonsense and offering visual explanations for potential object detection inaccuracies, leveraging labeled data contributed by domain experts. While the paper demonstrates its proficiency in identifying object detection errors using spatial commonsense, it does not present precise quantitative performance metrics for the CSK-SNIFFER system. In this study, the commonsense knowledge is given as default by domain experts, as mentioned before, unlike our methodology which extracts the information using an automated process. In addition, our proposed approach does not rely on any human intervention.

Chapter 4

Detecting $partOf$ Relations

The primary challenge addressed in this work revolves around the accurate identification and understanding of $partOf$ relationships between objects that have been detected within an image. In simpler terms, it involves understanding how various elements within the visual scene are interconnected, particularly when one object is a constituent or component of another. To elaborate further, consider the image shown in Figure 1.1. An example of a possible output from an object detection tool is visualized in Figure 1.1 (top). The input image is annotated with four objects, whose labels (“Person”, “Bicycle”, “Hand”) and bounding boxes are shown in the figure.

Utilizing an object detector's output (represented as $hasObject$ tuples), our framework's goal is to specify $partOf$ relationships between the detected object parts and the corresponding whole objects. Following our running example of Figure 1.1, our goal is to specify that each of the two bounding boxes labeled as “Hand”, is a part of the bounding box with label “Person”. However, a $partOf$ relationship should not be returned between the object parts “Hand” and the whole object “Bicycle”.

4.1 Problem Description

The problem mentioned above can be informally described as follows. Given a set of images, each containing a variety of objects detected by some object detection model, our goal is to specify the $partOf$ relations amongst the detected objects.

In this work, we differentiate whole objects and (object) parts, although both categories are in general mentioned as “objects” in the object detection literature and both categories use the same definition that we mentioned in Chapter 2 (i.e., a bounding box with a label). The dataset that has been used classifies labels into two distinct sets: a set of whole objects $\mathcal{O}$ and a set of object parts $\mathcal{P}$. We utilize this information in our approach, as is common in ML works. However, in our methodology where we gather semantic data from knowledge graphs related to these labels, the prior information is not essential.

We further consider a $partOf$ relation between a part and a whole object in a given image. A $partOf$ relation should be true, if factual information associating the corresponding p and o labels contained in the relation, i.e., a $meronym(p, o)$ relation, exists in the knowledge domain (a necessary, but not sufficient condition).

A formal definition of the *visual object-part discovery setting* is provided below:

Definition 4.1.1. A *visual object-part discovery setting* $\mathcal{V} = \langle \mathcal{I}, \mathcal{O}, \mathcal{P}, hasObject, partOf \rangle$ consists of

- a set of images $\mathcal{I}$,
- a set of whole-object labels $\mathcal{O}$,
- a set of object-part labels $\mathcal{P}$,
- a $hasObject \subseteq \mathcal{I} \times (\mathcal{P} \cup \mathcal{O}) \times BB$ relation, denoting the label and bounding box of an object within an image, where $BB \subseteq \mathbb{N}_0^4$, and
- a $partOf \subseteq \mathcal{I} \times \mathcal{P} \times BB \times \mathcal{O} \times BB$ relation, associating an object part with its whole object within an image, such that whenever $partOf(i, p, bb_1, o, bb_2)$ holds, it is necessary that both $hasObject(i, p, bb_1)$ and $hasObject(i, o, bb_2)$ also hold.

The *visual object-part discovery problem*, associated with this setting is the following:

Definition 4.1.2. Given an instance $\mathcal{V}^* = \langle \mathcal{I}, \mathcal{O}, \mathcal{P},\ hasObject^*,\ partOf^* \rangle$ of a visual object-part discovery setting, which we will call the ground truth, and a percentage threshold th, a *visual object-part discovery problem* is the optimization problem to find an instance $\widehat{\mathcal{V}} = \langle \mathcal{I}, \mathcal{O}, \mathcal{P},\ \widehat{hasObject}, \widehat{partOf} \rangle$ that

- maximizes the times that if $partOf^*(i, p, bb_1, o, bb_2)$ holds, then $\widehat{partOf}(i, p, bb'_1, o, bb'_2)$ also holds
- minimizes the times that if $partOf^*(i, p, bb_1, o, bb_2)$ does not hold, then $\widehat{partOf}(i, p, bb'_1, o, bb'_2)$ holds, and
- $bb_1 \approx bb'_1$ and $bb_2 \approx bb'_2$, meaning that the area of overlap of the corresponding bounding boxes is above the given threshold th.

In other words, the goal is to find a methodology that produces a $partOf$ relation that approximates as closely as possible some ground truth. Note that the object detection relation ($\widehat{hasObject}$) of the proposed instance $\widehat{\mathcal{V}}$ is not necessarily equivalent with that ($hasObject^*$) of the ground truth; errors produced by the $\widehat{hasObject}$ relation are propagated in the predictions, making the problem of approximating the $partOf^*$ relation, even more difficult.

In practice, the evaluation of a solution to the visual object-part discovery problem is performed by only considering how close the returned $partOf$ instances are to a ground truth, e.g., with respect to popular evaluation measures like precision, recall, F1-score, accuracy, etc, by also allowing some acceptable offset in the returned bounding boxes compared to the respective bounding boxes in the ground truth.

Apparently, one can train a neural network on the given dataset to more accurately detect the objects of interest or even train a classifier to explicitly detect $partOf$ relations.

Our goal is to show that this popular, yet laborious, practice is not the only direction, especially when annotated data is not readily available. Even with a typical object detector and without any $partOf$ training data, proper application of declarative methods can still offer leverage in tackling visual tasks.

4.2 Overview of the Methodology

Figure 4.1 illustrates the key components and workflow of our methodology. Initially, we extract relevant information pertaining to the dataset's labels from a Knowledge Graph (KG) using the **Knowledge Filtering Module**. In our work, we use the Semantic Pascal-Part [8] dataset. Knowledge graphs, which organize information by connecting concepts and relationships in a network, are pivotal for our approach. For this purpose, we have chosen ConceptNet [36], a well-established, open-source knowledge graph that aligns with our project's data requirements. The **Knowledge Filtering Module** serves a dual purpose: it extracts triples composing multiple subgraphs for our dataset's labels from ConceptNet, and it also filters this information to eliminate potential noisy data. All the retrieved triples undergo a filtering process with the remaining information being stored locally in Neo4j [33]. These subgraphs constitute our framework's knowledge domain, a structured and comprehensive collection of subject-specific facts. Subsequently, these stored subgraphs are converted into ASP facts and integrated into the Reasoner.

For the object detection task, i.e., the generation of $hasObject$ tuples for the images of the dataset, we employ the YOLOv5 model [10]. The outputs of object detection are used as input for the **Reasoner**, too. For reasoning, we employ the NeurASP framework [42], which combines neural networks with Answer Set Programming (ASP) for joint logic-based reasoning and deep learning. Within the Reasoner, we have formulated the $meronym$ **ASP Enrichment Ruleset**, comprising numerous ASP inference rules aimed at enhancing our knowledge domain.

Finally, leveraging the enriched knowledge domain and the detected objects, the $partOf$ **Discovery Ruleset** uses logical inference to identify $partOf$ relationships between the objects deteted in the images. The framework's outputs are stored locally in a JSON file.

4.3 Components Analysis

4.3.1 Knowledge Filtering Module

ConceptNet is widely used as a source of problem-agnostic, commonsense knowledge for application in a wide range of domains. Its broad coverage comes with the cost of noisy and often erroneous data, a problem more-or-less met with most other online repositories. Many solutions have been proposed in the literature for ironing out noise from such KGs and for keeping only the data that are relevant to the problem at hand. Some of these approaches take into consideration topological relations among nodes whose relatedness is under investigation, e.g., the number of connecting paths (e.g., [46, 22]) or the number of common nodes that two nodes share (e.g., [6, 45]); others lay emphasis on the semantics

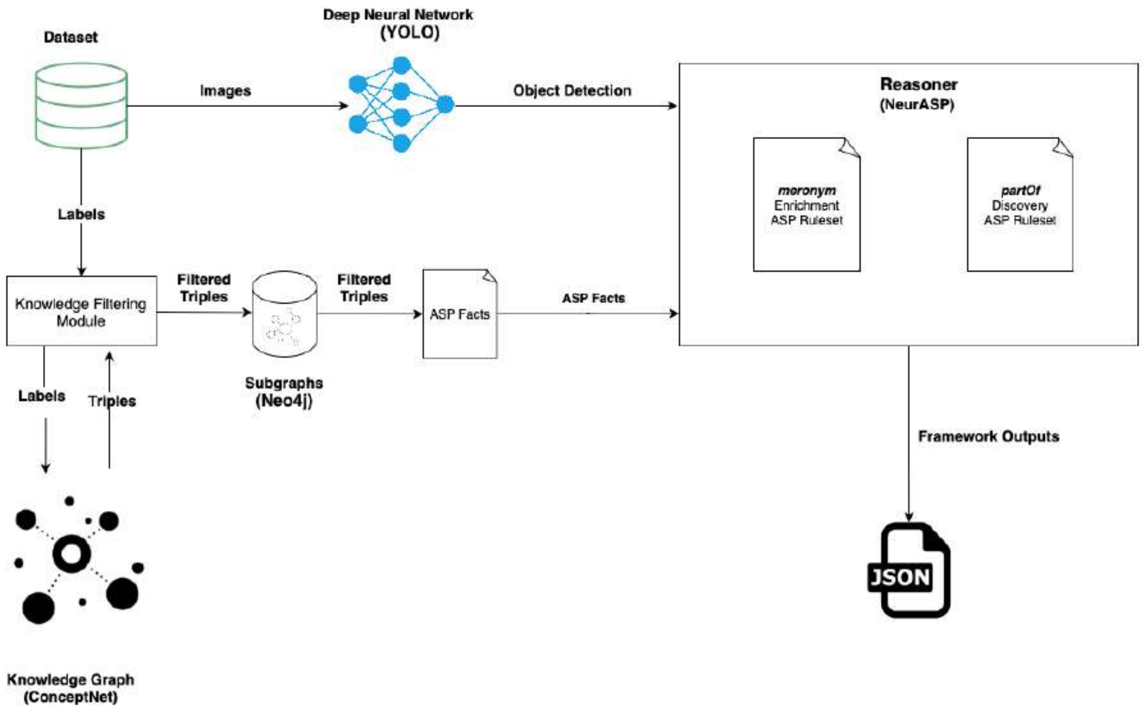


Figure 4.1: Component diagram of our proposed neuro-symbolic approach to $partOf$ relation detection in images.

(e.g., [28]). Unless one wishes to develop domain-specific solutions, no single line of knowledge extraction techniques has been shown to provide satisfactory results, as recent studies discuss [40].

In light of these results, we present below generic steps that consider both topological and semantics-related aspects, while being generic enough to easily be applied to similar KGs with minor adaptations.

The **Knowledge Filtering Module** begins by taking the dataset's labels as input in string format, retrieving all associated triples for specified relationships from ConceptNet. It then proceeds to apply a filtering process to eliminate potential noise, resulting in the creation of multiple subgraphs related to the dataset's labels. Once the filtering process is completed, the produced subgraphs are stored locally.

To assess the effectiveness of our approach, we consider as ground truth instance (i.e., $\mathcal{V}^*$) the Semantic Pascal-Part ontology, provided by the creators of the Semantic Pascal-Part dataset. Evidently, just by deploying the knowledge filtering module, we managed to successfully match a pretty low number of the "part of" relations (4/100) present in the ground truth.

Further details of the steps within the knowledge filtering module are presented below:

1. **Node Label Filtering.** Considering the labels of the objects we expect to find in the images, we extracted from ConceptNet a subgraph for each label, having as starting

node a ConceptNet node that can be associated to that label (exact string match or synonym). Moreover, we requested the starting node to be a noun. The resulting subgraph contains all 1-hop neighbors, which was sufficient for the given KG to get the information we needed for our purpose, although more distant neighbors may also be useful/needed in other KGs.

2. **Edge Label Filtering.** For the creation of the labels' subgraphs, we only kept the ConceptNet edges having one of the following labels: ***PartOf, HasA, IsA, HasContext, Synonym***. Note that, apart from being relevant to our purposes, similar relations are rather generic and can be met in other KGs, such as WordNet[32], Yago[37], DBpedia[1] etc.

3. **Edge Weight Filtering.** In ConceptNet, each edge is associated with a *weight*, denoting how strong the assertion is between the two linked nodes. As per ConceptNet's documentation, a weight of 1 is considered the "typical" weight, indicating that the assertion is as reliable as the majority of other assertions in the knowledge graph. Hence, considering the prior statement, we pruned edges with weight lower than 1, for more reliable information.

4. **Node Type Filtering.** Finally, we applied another filtering step, based on the node type capturing the context of use. For example, one can find in ConceptNet both of the following triples *(cat, **is type of,** animal)* and *(cat, **is type of,** woman)*. Although such associations are not necessarily erroneous, the latter is out of context for our purposes. In order to keep context-specific data in our subgraphs, we exploited the characterization (*sense label*) that ConceptNet assigns to nodes according to their category (e.g., animal, person, artifact). As such, although in ConceptNet both "cat$^{(\text{n, animal})}$" and "cat$^{(\text{n, person})}$" can be found, we only considered the former for our purposes. This information is readily available in ConceptNet for most nodes, but can also be easily extracted with other methods, e.g., by considering WordNet inheritance relations, if for instance one utilizes KGs that lack such characterizations.

Eventually, we extracted a subset of ConceptNet comprising 6,544 nodes and 7,617 edges and stored it locally in a Neo4j Graph Database. Figure 4.2 provides a visual representation of the subgraph constructed via ConceptNet for the label "Bird". In this illustration, we have chosen to display only the edges labeled "Part of" and "IsA". In the subgraph numerous triples exist, such as (pennon, isA, wing), (syrinx,is part of ,bird), etc. Notably, the only triple found in our ground truth and directly derived from ConceptNet is (beak, is part of, bird), while others can be indirectly obtained (e.g., head, tail). However, some noisy cases are also present in the subgraph. For instance, there is a triple indicating the relationship (bird, is part of, bird). Without appropriate handling, such data could potentially lead to erroneous outputs.

This information will be used by the reasoning module described later, in order to help identify relevant relations.

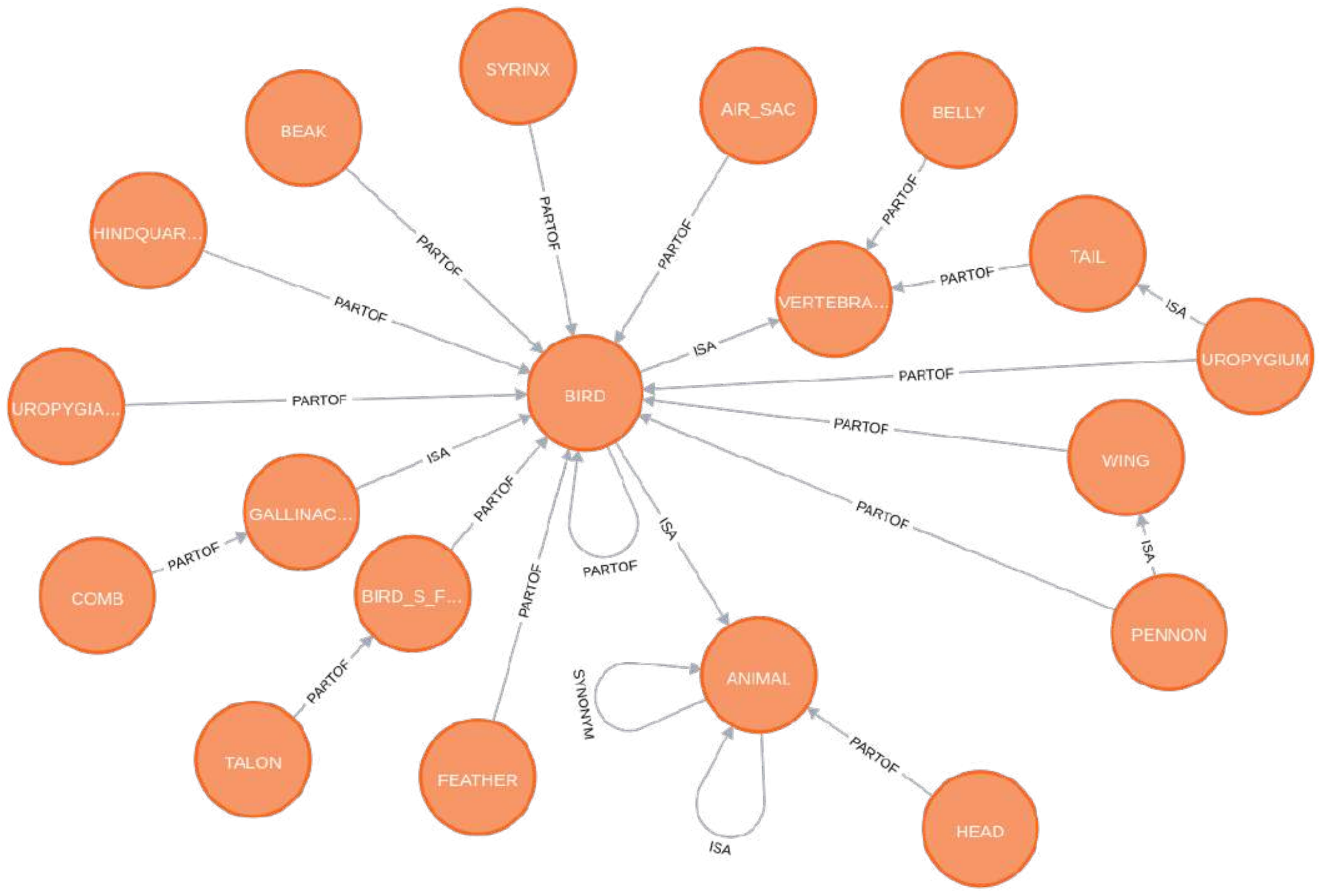


Figure 4.2: Subgraph with the extracted information from ConceptNet regarding the label "Bird". In the subgraph, a number of relations connected with "Bird" are shown. For instance, the triples (wing,partOf,bird), (beak,partOf,bird), (bird,isA,animal), etc., exist in the subgraph.

4.3.2 Object Detection Model

For the detection of the visual information that exists in an image, i.e., the $hasObject$ tuples, we rely on typical off-the-shelf tools and methods. In particular, we use a YOLOv5l [10] neural network, which we fine-tuned on the Pascal-Part Dataset for object detection. The neural network was fine-tuned using three parameter variations. We fine-tuned:

1. A model using pre-trained weights (PreYoloL)
2. A model from scratch (ScrYoloL)
3. A model using noise-imputed annotations of the training set (NoisyYoloL)

The **object detection model** receives images as input and provides output in the form of detected objects, along with their associated bounding box coordinates and corresponding class labels. An example of the outputs of the object detection model is visualized in Figure 4.3. Given as input the image shown on the left, the object detection model outputs the coordinates of the illustrated bounding boxes coupled with a classification label, as presented on the right image. Note that not all the outputs of the object detector were visualized in the example, for clarity reasons.

The outputs of the object detection model are given as input to the NeurASP framework.

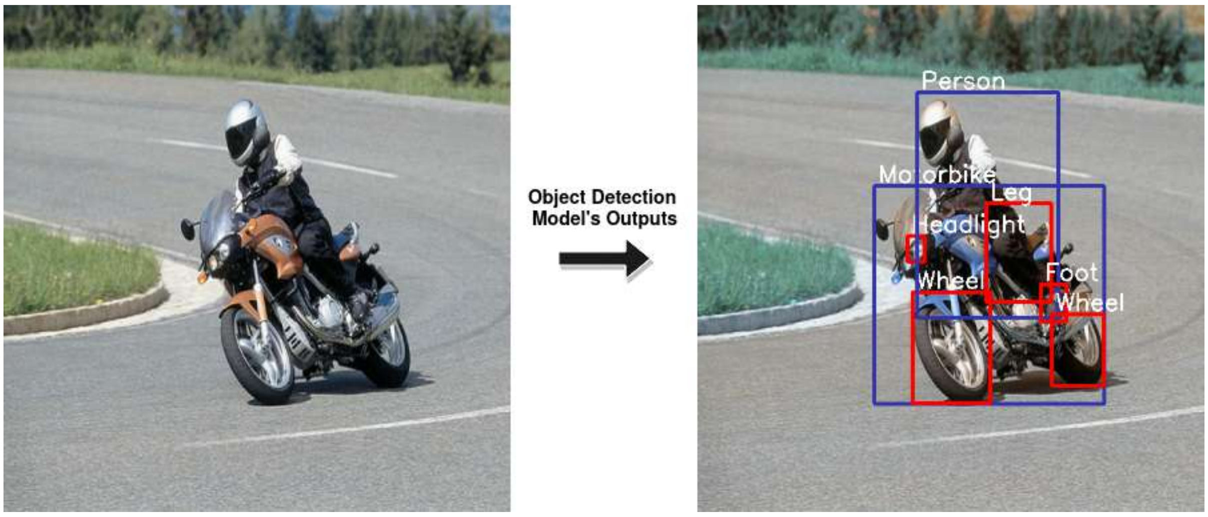


Figure 4.3: Example of an object detector's input (left) and output (right).

4.3.3 Reasoning Module

For the reasoning task, we deployed the NeurASP framework, which combines neural networks and ASP. The reasoner takes as input (i) the information extracted from ConceptNet, stored in form of subgraphs and (ii) the objects detected by the object detection model in an image ($hasObject$ relations). The reasoner's role is to enhance the information within the subgraphs, i.e., the knowledge domain, using the $meronym$ Enrichment Ruleset. This enrichment aims to improve the matching ratio of the $meronym$ relationships with respect to the ground truth. By combining the enriched knowledge domain with the object detector's outputs and employing the $partOf$ Discovery Ruleset, the reasoning module identifies $partOf$ relationships between the detected objects. The reasoning process is explained below in more details.

To start with, all information extracted from the KG is then modeled into ASP facts, in order to be used for the inference purposes stated above. Specifically, with Cypher [13] queries (shown in Appendix B), we extracted the stored data and transformed them into ASP predicates. For example, the ASP fact $meronym(beak, bird, animal)$ states that a $beak$ is part of a $bird$, and both concepts are categorized as $animal$. Note that, although typically $meronym$ can be considered a binary relation, we extended it with a category argument (e.g., animal), to differentiate between object and part labels that may appear under various categories.

In addition, we manually characterized each label according to its associated category, e.g., $object(car,\ artifact), part(tail,animal)$. These label categories were determined through diligent searching within ConceptNet. Furthermore, we separated the labels that denote objects to the ones that denote object parts. The object detector's outputs are also represented in ASP within the NeurASP framework.

When all of the aforementioned information is successfully formulated in ASP, we apply a number of ASP rulesets which eventually can lead to the generation of $partOf$ relations. We model 3 rulesets, namely, the *bounding box overlap*, the *knowledge* $(meronym)$ *enrichment*, and the $partOf$ *detection* ruleset, each discussed in the sequel.

First, though, let us briefly describe how NeurASP integrates neural network estimations as part of the logic program. In NeurASP, a neural network is represented by a *neural atom* nn, which in our case takes the form

$$nn(label(1, I, B),[person, cat, \ldots, other]) :- \\ box(I, B, X_{\text{min}}, Y_{\text{min}}, X_{\text{max}}, Y_{\text{max}}, C).$$

This rule classifies the bounding box B, which is located at $(X_{\text{min}}, Y_{\text{min}}, X_{\text{max}}, Y_{\text{max}})$ within image I, into one (1) of the classes (labels) {person, cat, ..., other}, with confidence C. As such, for each image, the neural network outputs labels to each bounding box found, which is then used as input for the inference rulesets described next.

For instance, NeurASP converts the outputs of the deployed neural network in ASP for the image in Figure 4.3 as follows.

$box(img, b0, 223, 228, 513, 503, 958).$
$box(img, b1, 265, 291, 286, 324, 472).$
$box(img, b2, 271, 362, 369, 502, 772).$
$box(img, b3, 277, 110, 455, 394, 930).$
$box(img, b9, 364, 250, 446, 374, 891).$
$box(img, b10, 433, 352, 465, 399, 825).$
$box(img, b11, 447, 390, 512, 480, 796).$

Subsequently, the *neural atom* produces atoms regarding the labels of the bounding boxes.

$label(0, img, b0, motorbike).$
$label(0, img, b1, headlight).$
$label(0, img, b2, wheel).$
$label(0, img, b3, person).$
$label(0, img, b9, leg).$
$label(0, img, b10, foot).$
$label(0, img, b11, wheel).$

Bounding Box Overlap Ruleset. Given a set of labeled bounding boxes identified in an image and transformed into ASP atoms by NeurASP, a typical step is to identify the portion of overlap between pairs of bounding boxes. In fact, very often this is the only measure applied to infer $partOf$ relations [9]; this is going to be a baseline case to compare our method against, as explained in the next section.

For our purposes, we devised a set of ASP rules, that are presented in Appendix A.1, to enable our framework to discover the spatial relations between the detected objects in an image. Particularly, we wish to find pairs of bounding boxes with 75% overlap (empirically set); if such coverage is identified, we store this information with the help of the $candidatePartOf/2$ predicate, which takes the IDs of the two bounding boxes as arguments. Specifically, a $candidatePartOf(b1, b2)$ atom denotes that the area of bounding box $b2$ covers by 75% or more the area of $b1$, implying that $b1$ may be part of $b2$.

Consider again the example image shown in Figure 4.3. The prior ruleset will infer the following atoms for the objects detected in the image.

$candidatePartOf(b1, b0).$
$candidatePartOf(b2, b0).$
$candidatePartOf(b9, b0).$
$candidatePartOf(b10, b0).$
$candidatePartOf(b11, b0).$
$candidatePartOf(b1, b3).$
$candidatePartOf(b9, b3).$
$candidatePartOf(b10, b3).$

Depending solely on the visual data, an approach would erroneously identify the aforementioned atoms as the solution to the target problem. Evidently, such an approach would yield incorrect results and poor performance. For example, resulting to the conclusion that an object part labeled as “headlight”, is part of a whole object “person”, contradicts commonsense knowledge.

Knowledge ($meronym$) Enrichment Ruleset. As a next step, we materialize the implicit knowledge that exists in the extracted graph from ConceptNet and enrich it, by exploiting the synonym, inverse, subsumption, and meronym relationships between concepts. We therefore apply a set of logical inference rules, as for instance:

$$\begin{aligned}
&meronym(X, Z, C) :- \\
&\qquad\qquad meronym(X, Y, C), \\
&\qquad\qquad isSynonymWith(Y, Z, C). \\
&meronym(X, Y, C) :- \\
&\qquad\qquad meronym(X, Z, C), \\
&\qquad\qquad hypernym(Z, Y, C).
\end{aligned}$$

The first rule states that if we know that label X is a meronym of label Y, and Y is a synonym of Z, then we can infer that X should also be considered as a meronym of Z.

The second rule indicates that if we know that label X is a meronym of label Z, and Z is a hypernym of Y, then we can deduce a $meronym$ relationship between X and Y. A *hypernym* is a term that represents a broader or more general category of concepts or words. The complete ruleset is showcased in the Appendix A.2.

Revisiting the subgraph of the label "bird", illustrated in Figure 4.2, our approach, utilizing the second $meronym$ rule, can make a logical inference, establishing the fact $meronym(head, bird, animal)$, and consequently, it improves the matching ratio with the ground truth. This inference stems from the presence of the following information in our knowledge domain: $(animal, hypernym, bird)$ and $(head, meronym, animal)$. Note that all these labels share the common category of "animal".

Another characteristic example is the "aeroplane" label found in the dataset, which is poorly described in ConceptNet, e.g., no association with the "wheel" concept is modeled. Yet, such an association does exist for the "airplane" ConceptNet node; by exploiting the synonymity relation between these two terms, we can significantly enrich the information obtained.

It is interesting to note that by relying exclusively on the $PartOf$ relation found in ConceptNet, i.e., without applying the knowledge enrichment ruleset described above, we would only match information about 4 out of the 100 $meronym$ relations that exist in the Semantic Pascal-Part ontology, which acts as our ground truth. Yet, with the application of such generic inference rules, we achieved a 85% coverage of the $meronym$ relations present in the ground truth.

***PartOf* Detection Ruleset.** We exploit all the information obtained by the previous steps, in order to detect $partOf$ relations between objects detected in an image. One can devise different inference schemes, from rather simplistic to more complex ones, as we elaborate in the next section. A declarative approach, as the one we propose in this study, enables not only intuitiveness in modeling the inference rules that generate $partOf$ knowledge, but also interpretability of the results, in order to promote explainability and facilitate future refinements.

For our purposes, in addition to a couple of baselines that only consider the overlap between bounding boxes or the knowledge that is explicitly found in ConceptNet (without any enrichment), we further suggest a simple, yet effective scheme, that can be summarized by the following rule:

$$\begin{aligned}
&partOf(B1, L1, B2, L2) :- \\
&\qquad candidatePartOf(B1, B2), \\
&\qquad partBox(B1, L1, C), \\
&\qquad objectBox(B2, L2, C), \\
&\qquad meronym(L1, L2, C).
\end{aligned}$$

Intuitively, a bounding box $B1$ having label $L1$ is part of bounding box $B2$ with label $L2$ if (i) the inspected bounding boxes are overlapping with a ratio over 75%, i.e., $candidatePartOf(B1, B2)$, (ii) box $B1$ is of type $Part$, box $B2$ is of type $Object$, and both belong to the same category C, and (iii) a $meronym$ association between the detected labels has been extracted from ConceptNet or inferred using inference rules. A number of auxiliary rules are modeled in our logic program, in order to generate the predicates mentioned in the aforementioned rule.

We exhibit how our $partOf$ detection ruleset effectively addresses the issue discussed in the preceding paragraphs, specifically concerning the example image depicted in Figure 4.3. Our methodology filters the $candidatePartOf/2$ atoms using the auxiliary predicates that were produced, along with the knowledge domain which is established through ConceptNet and enriched by the $meronym$ Enrichment ruleset. By excluding any conflicting associations that go against commonsense knowledge, such as inferring that the "headlight" is a part of the "person" when considering only spatial information in our example, this straightforward yet powerful approach yields the following inferences.

$partOf(b1, headlight, b0, motorbike).$
$partOf(b2, wheel, b0, motorbike).$
$partOf(b11, wheel, b0, motorbike).$
$partOf(b9, leg, b3, person).$
$partOf(b10, foot, b3, person).$

One can observe that our ruleset dropped the outputs that don't align with our knowledge domain, in contrast to what a method focusing only on visual data and spatial associations would produce.

4.4 Example

Consider the image shown in Figure 1.1 (top). The output of the NN would generate the following facts:

$box(img, b1, 63, 433, 411, 624, 832).$
$box(img, b3, 160, 301, 346, 611, 932).$
$box(img, b17, 267, 439, 305, 464, 795).$
$box(img, b21, 314, 427, 348, 451, 862).$

Next, the *neural atom* would produce the facts regarding the label of the bounding boxes:

$label(0, img, b1, bicycle).$
$label(0, img, b3, person).$
$label(0, img, b17, hand).$
$label(0, img, b21, hand).$

Considering the type and category of every label, further information is constructed:

$objectBox(b1, bicycle, artifact).$
$objectBox(b3, person, body).$

$partBox(b17, hand, body).$
$partBox(b21, hand, body).$

One can observe in the image that the bounding boxes indicating the "Hands" are both covered enough by the bounding boxes of the "Bicycle" and the "Person", so the following facts are generated, as well:

$candidatePartOf(b17, b1).$
$candidatePartOf(b21, b1).$
$candidatePartOf(b17, b3).$
$candidatePartOf(b21, b3).$

Taking advantage of the knowledge we acquired through the previous steps, since the fact $meronym(hand, bicycle, artifact)$ does not exist in our knowledge base, the candidate relations associating the two labels are rejected. The output of our methodology would be:

$partOf(b17, hand, b3, person).$
$partOf(b21, hand, b3, person).$

4.5 Summary

The methodology presented here addresses the well-known challenge in Computer Vision, which involves detecting $partOf$ relationships among objects in an image. By leveraging an open knowledge graph like ConceptNet to gather information about the labels in the dataset and enhancing this information with a range of generic inference rules, we achieved an 85% matching coverage of $meronym$ relations as found in the ground truth. However, achieving a perfect 100% coverage has proven challenging due to the potential introduction of noisy information, despite our efforts to minimize its impact and maintain result quality.

Furthermore, by employing our $partOf$ Discovery ruleset, our methodology successfully identified $partOf$ relationships among detected objects with a high level of precision, filtering out incorrect associations that baseline approaches might produce. The results presented in Chapter 7 affirm that this approach outperforms baseline methods that rely solely on visual data.

It's significant to highlight that these devised rules can be easily adapted for various tasks. More elaborate inference schemes can be deployed, especially if one aims to integrate domain-specific knowledge or apply inductive logic programming to create rules tailored to the data. However, the key takeaway from this study is that even with a simple yet generic approach, the combination of data-driven methods with external knowledge and inference can lead to improved performance in detection tasks.

Chapter 5

Identifying Errors in Object Detector's Output

In alignment with the previously outlined challenge, which centers on defining $partOf$ relationships among detected objects, this investigation pivots towards the task of identifying errors in the outputs of object detection models.

Despite their considerable capabilities, object detection models are not immune to errors, particularly in complex and dynamic real-world situations, or when training data are sparse or noisy. The utilization of external knowledge sources and commonsense reasoning can be proven quite valuable in detecting potential model errors. These resources provide a valuable mechanism for flagging incorrect outputs by cross-referencing the model's results with established real-world knowledge.

By detecting errors, we provide a critical first step in quality control, allowing subsequent processes to further investigate and resolve the identified errors. While the primary objective is error detection, our aim extends beyond mere identification. We aspire to address these errors effectively, whether by providing insights, generating corrective actions, or enhancing the accuracy of the object detection outcomes. This separation of detection and resolution ensures a more systematic and efficient approach to improving the accuracy and reliability of object detection models, making them more adaptable to diverse and dynamic real-world scenarios.

5.1 Problem Description

The issue we previously outlined can be informally expressed in the following way. Given a set of images and the outputs of an off-the-shelf object detection model for this set of images, our objective is to pinpoint cases where the object detection model generates incorrect or incomplete results. These errors may include misclassifications, missed objects, or incorrect bounding boxes, according to a provided ground truth of object detection results for the input images.

The prior problem can be formally defined, by adapting Definition 4.1.2, as follows:

Definition 5.1.1. Given an instance $\mathcal{V}^* = \langle \mathcal{I}, \mathcal{O}, \mathcal{P}, \, hasObject^*, \, partOf^* \rangle$ of a visual object-part discovery setting, which we consider the ground truth, and another instance $\widehat{\mathcal{V}} = \langle \mathcal{I}, \mathcal{O}, \mathcal{P}, \, \widehat{hasObject}, \, \widehat{partOf} \rangle$, where $\widehat{hasObject}$ approximates the $hasObject^*$ relation, a *simplified error detection* problem concerns the finding of erroneous cases of $\widehat{hasObject}$ relations, in terms of their labels or the coordinates of their bounding boxes.

5.2 Overview of the Methodology

In Figure 5.1, we present an overview of our Error Detection methodology. We extend the existing $partOf$ methodology to include the capability to identify potential errors that could arise in the object detection model's outputs. The process leading up to the $partOf$ Discovery is identical to the one detailed in Section 4.2. However, an additional ruleset was formulated, the **Error Detection Ruleset**, to tackle the aforementioned task of error identification. The **Error Detection Ruleset** is in charge of discovering potential errors in the object detector's outputs, categorizing them and reporting which detection arose the error. Once errors are detected, the **Error Handling Module** takes charge of resolving these errors based on their specific category.

Throughout the error handling process, some data may undergo modifications. Consequently, the outputs of these processes are fed back into the error detection module to scan for potential new error triggers. This iterative process between the modules continues until no more errors are identified. We store the outputs of the framework locally in separate JSON files. Upon the completion of the execution, two JSON files are generated: one containing the detected objects and their $partOf$ relations, and another containing the identified errors.

5.3 Error Detection and Handling

This section covers the rules and techniques that were implemented to expand our framework's capabilities to identify errors in the outputs of the object detection model and take actions to rectify them.

Observing the outputs of the object detection model ($hasObject$ tuples) and the prior rulesets, we were led to the assumption that by combining the two, we could possibly discover erroneous outputs of the model.

Initially, we attempted to address this issue by formulating some constraint rules, led by the capabilities offered by ASP. The constraint rules that were devised were trying to detect and resolve the errors simultaneously. This approach had a negative effect in terms of the mAP score, which is a metric that is traditionally used in Computer Vision to assess an object detection model. (More detailed explanation in 7.2.1). Our initial intention was to improve the object detection model's performance, and the reported results were doing the exact opposite. Monitoring the results to understand why that happened, we realised that we were detecting errors quite naively, disregarding important information that could be attained by the existing outputs of the framework. Furthermore, we came to the

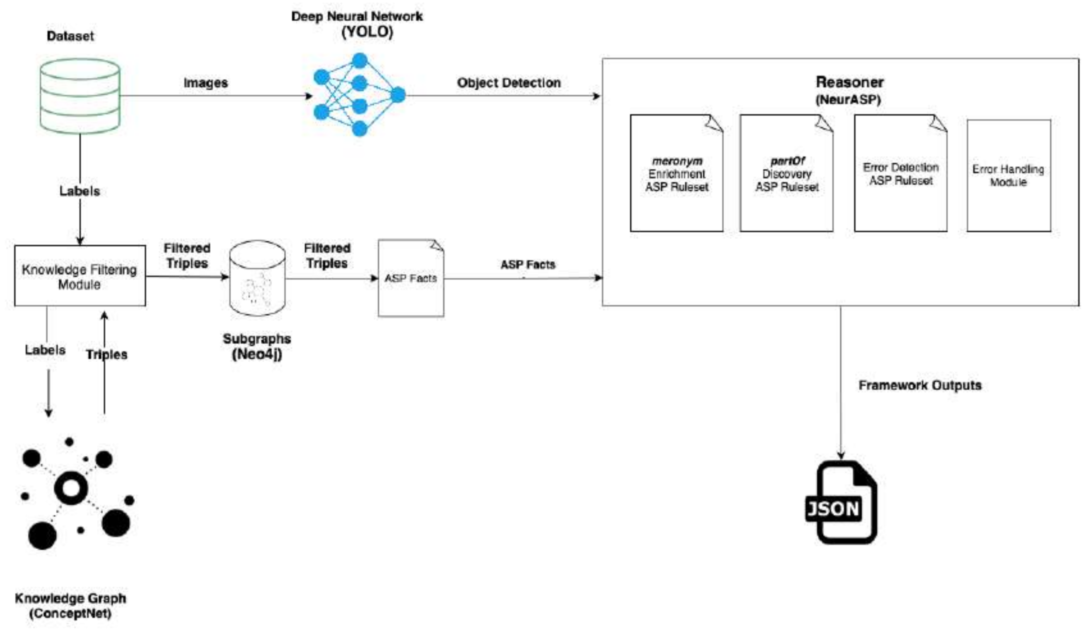


Figure 5.1: Error detection and Handling methodology outline.

conclusion that there are different cases of errors, that can be detected by different conditions, and must be resolved by different techniques. Hence, this called for the separation of the error detection and the error handling procedures. The errors can be categorized as presented in Table 5.1 and explained in detail in the next sections.

Table 5.1: Error Cases.

Error Case	Erroneous Detection	
	Object Part	Whole Object
Case P	✓	
Case O		✓
Case PO	✓	✓
Case MO		✓

Beyond error detection, the framework extends to proposing a methodology for handling identified errors based on their types. This involves potential modifications to the object detection model's outputs or providing insights to facilitate manual correction of detected errors.

5.3.1 Error Case P

Case P errors indicate that an object part that is present in the detector's outputs, was possibly misclassified.

An Error Case P occurs when an object part p is detected (i) with low confidence score, (ii) can be spatially related with a whole object o and (iii) lacks semantically association with any whole object present in the image, then there is an error associated with the object part p.

5.3.1.1 Error Detection Methodology

Error case P can be formulated in ASP as shown below. Numerous auxiliary rules were devised to establish the presented rule. The $spatialPartOf/4$ rule defines a spatial connection between bounding boxes $B1$ and $B2$ considering their coordinates and their type (*Object/Part*). The $highConfidence/1$ predicate specifies that the confidence score of a bounding box B exceeds the selected threshold, which was empirically set to 0.4. The $single/1$ predicate indicates that an object part is not semantically associated with any whole object within the image, while the predicates $knownPart/1$ and $knownObject/1$ report that in our knowledge domain there is $meronym$ information regarding the labels of the associated bounding boxes. These latter predicates were formulated to disregard labels for which our knowledge domain lacks any relevant information. The complete collection of the auxiliary rules that were formulated, can be found in Appendix A.4.

$$
\begin{aligned}
&errorCaseP(B1) :- \\
&\qquad spatialPartOf(B1, \_, B2, \_), \\
&\qquad knownPart(B1), \\
&\qquad knownObject(B2), \\
&\qquad highConfidence(B2), \\
&\qquad not\ highConfidence(B1), \\
&\qquad single(B1).
\end{aligned}
$$

An example of an identified Error Case P is visualized in Figure 5.2. In the image, the object detector identified a "Handlebar" (depicted by the yellow box) with low confidence. While it was spatially linked to a "Person" (indicated by the blue box), there was no semantic correlation with any whole object in the image. Evidently, the framework correctly detected this error.

5.3.1.2 Error Handling Methodology

While the framework is being executed, not only it detects objects using the model but also stores the class probability distribution for each bounding box. Consequently, for every bounding box identified in an image, we maintain a stack that holds its top k most likely classes based on the model's predictions.

As already mentioned, errors falling under Case P indicate potential misclassification of an object part. In an effort to rectify such errors and enhance the object detection performance, we undertake a procedure where we substitute the label of the object part with the subsequent most probable label from the stack. This iterative process continues

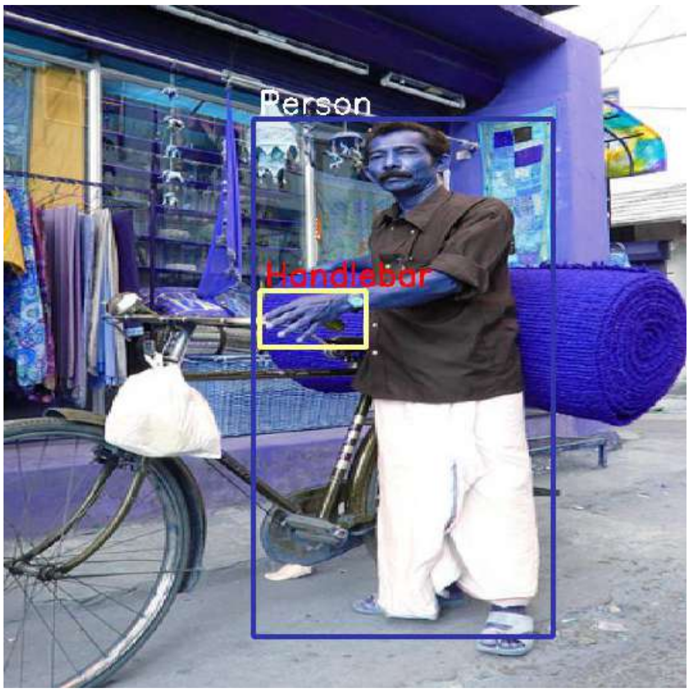


Figure 5.2: Error Case P example. In the image, an object part classified as “Handlebar” is detected with low confidence, that can be spatially connected with the detected whole object “Person”. Because the two cannot be semantically associated, the framework identifies a case P error triggered by the “Handlebar” bounding box.

until the specific bounding box no longer generates errors in the framework’s inferences. It is important to note that if the probability of a class in the stack is equal to zero (0) or if the stack for a particular bounding box is empty, that box is categorized as “Other”.

In Figure 5.3, we showcase how our proposed technique handles the error that was identified in the earlier example. The image on the right demonstrates the outcomes after error resolution. The framework adeptly adjusts the label from “Handlebar” to “Hand”, aligning it accurately with the correct classification.

5.3.2 Error Case O

Case O errors indicate that the object detection model has possibly returned incorrect outputs about a whole object detection.

This error case occurs when (i) a whole object o is detected with low confidence score, (ii) o can be spatially related with an object part p with high confidence, (iii) o and p cannot be semantically associated, and (iv) the semantically associated parts with the whole object o do not exceed the defined threshold.

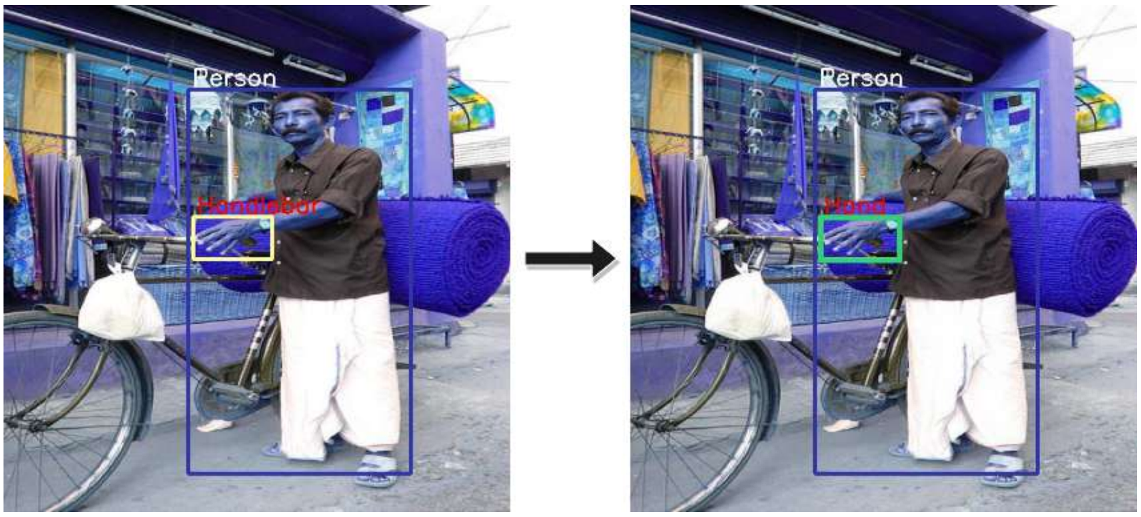


Figure 5.3: Error Case P handling example. The label "Handlebar" of the bounding box that triggered the error is correctly changed to "Hand".

5.3.2.1 Error Detection Methodology

Following the ASP rule that was devised to capture Case O errors is displayed. While the rest of the rules were already explained in 5.3.1.1, a new predicate is introduced in this rule. A $context/1$ fact defines that a bounding box classified as a whole object is semantically linked to a certain number of object parts, denoted as N, each with a high confidence score. The specific value of N has been set as 2.

$$
\begin{aligned}
&errorCaseO(B2) :- \\
&\qquad spatialPartOf(B1, \_, B2, \_), \\
&\qquad knownPart(B1), \\
&\qquad knownObject(B2), \\
&\qquad highConfidence(B1), \\
&\qquad not\ highConfidence(B2), \\
&\qquad not\ context(B2), \\
&\qquad single(B1).
\end{aligned}
$$

Figure 5.4 exhibits the occurrence of a Case O error. Here, "Animal Wing" object parts (highlighted by the blue boxes) have been detected with high confidence, and they are spatially associated with a low-confidence "Aeroplane" (indicated by the orange box). This contradicts the commonsense knowledge present in our knowledge domain, leading to the appropriate identification of a Case O error (indicating that the "Aeroplane" box is probably wrong).

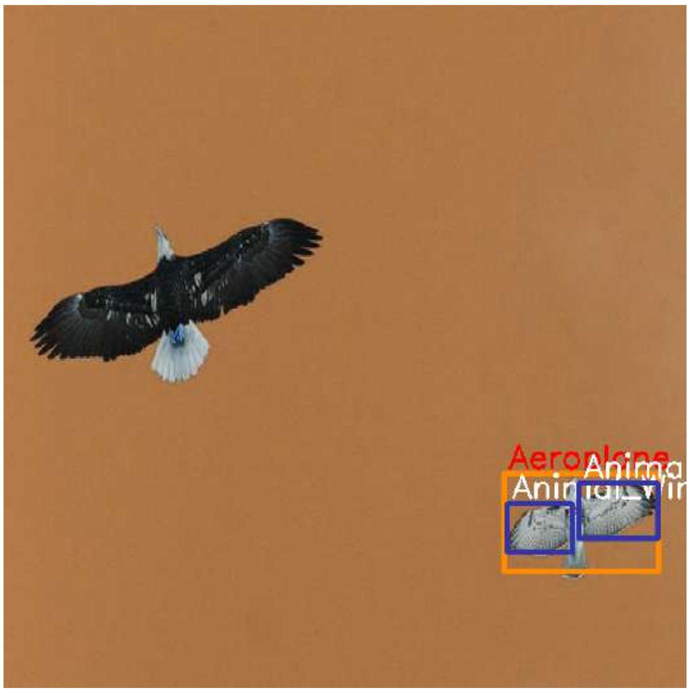


Figure 5.4: Error Case O example. The error is triggered by the whole object “Aeroplane”. This happens because “Aeroplane” is detected with low confidence and is spatially associated with the bounding boxes labeled as “Animal wing”, which have been detected with high confidence.

5.3.2.2 Error Handling Methodology

As it was pointed out, Case O errors indicate that the object detector possibly misclassified a whole object. This kind of errors are handled similarly as Case P errors by updating the whole object’s label following the procedure that was previously mentioned in Section 5.3.1.2.

The image on the right of Figure 5.5 exemplifies how the framework rectifies the error mentioned in the previous section, by changing the label from “Aeroplane” to “Bird”.

5.3.3 Error Case PO

This kind of error indicates that either an object part p or a whole object o was wrongly detected.

Error Case PO involving a detected whole object o and a detected object part p occurs when (i) p and o are both detected with low confidence score, (ii) p and o can be spatially related, (iii) p and o cannot be semantically associated and (iv) there are not enough semantically associated parts with the whole object o.

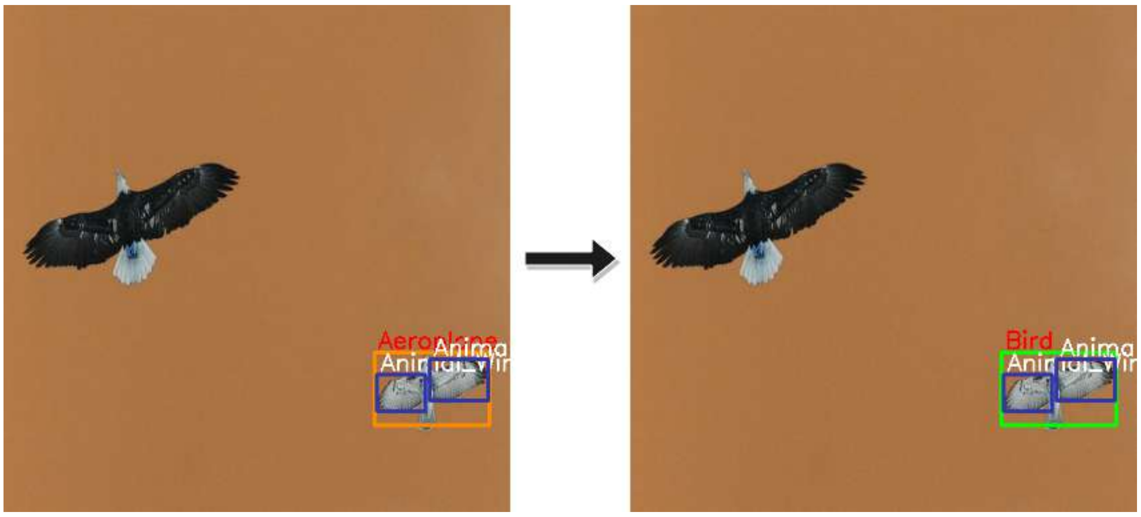


Figure 5.5: Error Case O handling example. To resolve the error, the framework changes the label "Aeroplane" to "Bird".

5.3.3.1 Error Detection Methodology

The ASP formulation of the specified error is showcased below. The composing factors of the rule are already explained in the previous error cases. A Case PO error event is shown in Figure 5.6. The detection of "Person" and "Muzzle" (highlighted in purple) met the predefined conditions for Case PO errors. Clearly, both outputs are incorrect.

$$
\begin{aligned}
&errorCasePO(B1, B2) :- \\
&\qquad spatialPartOf(B1, \_, B2, \_), \\
&\qquad knownPart(B1), \\
&\qquad knownObject(B2), \\
&\qquad not\ highConfidence(B1), \\
&\qquad not\ highConfidence(B2), \\
&\qquad not\ context(B2), \\
&\qquad single(B1).
\end{aligned}
$$

5.3.3.2 Error Handling Methodology

This category of errors arises when either one or both of the entangled bounding boxes are inaccurately classified. Our framework handles this error case in a similar manner as discussed in 5.3.1.2; however, with a distinction in label replacement, guided by the probability distributions associated with both bounding boxes. It peeks into the stacks and summarizes the probabilities for the next two most probable label combinations, considering both altering the label of the object part while retaining the label of the whole object,

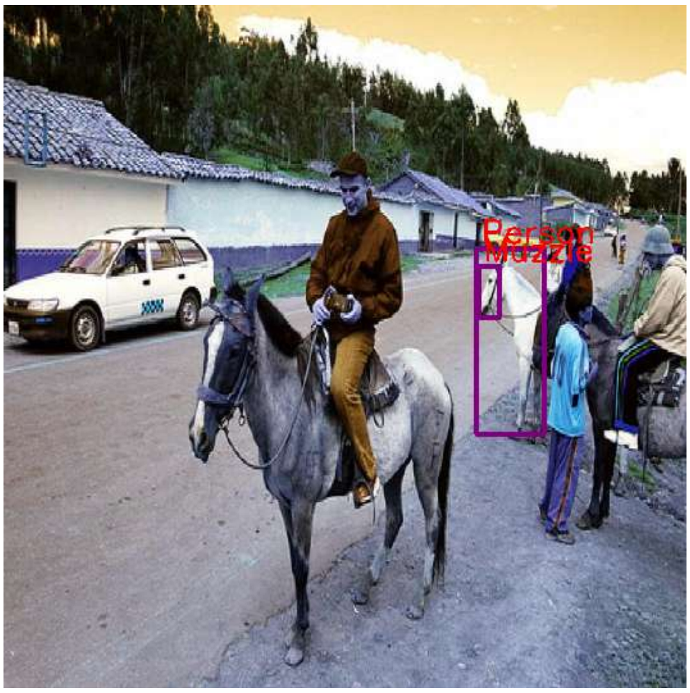


Figure 5.6: Error Case PO example. The detection of spatially connected "Person" and "Muzzle" objects, both with low confidence, led to the event of the case PO error.

and vice versa. Subsequently, the labels are adjusted to align with the label combination that yields the highest overall probability summation.

Following the error resolution process for the previously shown case, the framework addresses one of the problematic labels by changing the "Muzzle" label to "Head", as demonstrated in the image on the right of Figure 5.7.

5.3.4 Error Case MO

Error Case MO indicates a missing whole object (MO) from the object detector's outputs for the image. This kind of error can be triggered by various subcases.

- Subcase 1: A whole object o and an object part p are detected (i) p and o are both detected with high confidence score, (ii) p and o can be spatially related and (iii) the object part p cannot be semantically associated with any whole object detected in the image.

- Subcase 2: A whole object o and an object part p are detected, (i) the object part p has high confidence score, (ii) the whole object o is semantically associated with a number of high confident object parts, (iii) p and o can be spatially related but (iv) the object part p cannot be semantically associated with any whole object detected in the image.

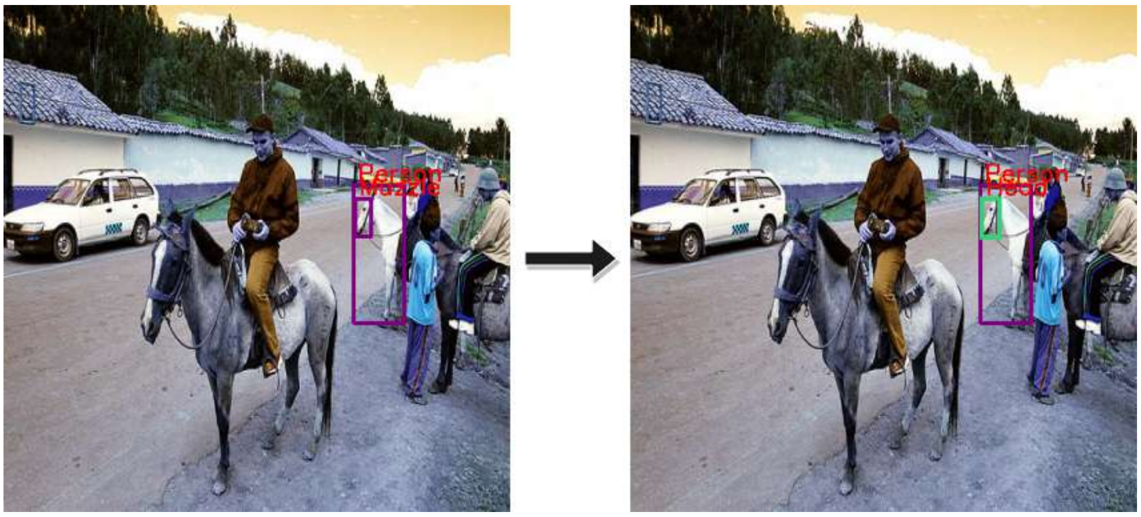


Figure 5.7: Error Case PO handling example. The framework addresses the error by changing the "Muzzle" label to "Head". Following the label modification, there is no violation within the knowledge domain. Consequently, the "Person" box, while visually incorrect, remains unalterable.

- Subcase 3: An object part p is detected (i) with high confidence score and (ii) it cannot be spatially associated with a whole object o.

5.3.4.1 Error Detection Methodology

Case MO errors are formulated in ASP as displayed next. Note that Subcases 1 and 2 can be combined into one rule when formulated in ASP using the disjunction rule $1\,\{highConfidence(B2); context(B2)\}$. This rule defines that if at least one of $highConfidence/1$ or $context/1$ holds, then the disjunction rule holds as well.

$$
\begin{aligned}
&errorCaseMO(B1) :- \\
&\qquad\qquad spatialPartOf(B1, \_, B2, \_), \\
&\qquad\qquad knownPart(B1), \\
&\qquad\qquad knownObject(B2), \\
&\qquad\qquad highConfidence(B1), \\
&\qquad\qquad 1\,\{highConfidence(B2); context(B2)\}, \\
&\qquad\qquad single(B1).
\end{aligned}
$$

$$
\begin{aligned}
&errorCaseMO(B1) :- \\
&\qquad\qquad not\ spatialPartOf(B1, \_, \_, \_), \\
&\qquad\qquad knownPart(B1), \\
&\qquad\qquad highConfidence(B1), \\
&\qquad\qquad single(B1).
\end{aligned}
$$

Undetected whole object error correctly occurs in Figures 5.8 and 5.9. Numerous object parts with high confidence were identified, but were unable to semantically correlated with a whole object.

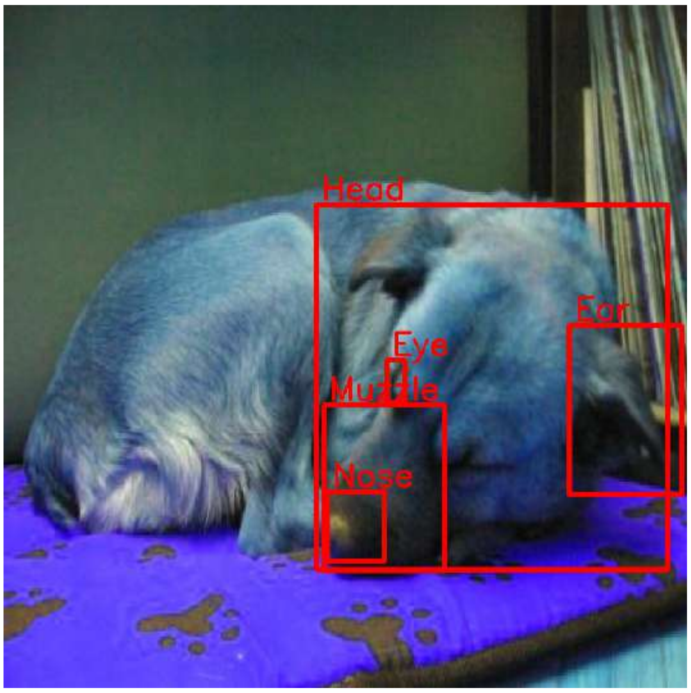


Figure 5.8: Error Case MO example 1. Multiple high-confidence object parts are detected (e.g., “Nose”,“Muzzle”, etc.) that cannot be spatially connected with any whole object within the image.

5.3.4.2 Error Handling Methodology

The current challenge extends beyond merely recognizing the absence of a whole object; it involves quantifying the number of missing detections within that category and providing a list of potential classes for those missing objects.

To address this challenge, a new set of rules has been developed. This new ruleset takes on the responsibility of identifying potential missing whole object classes. It

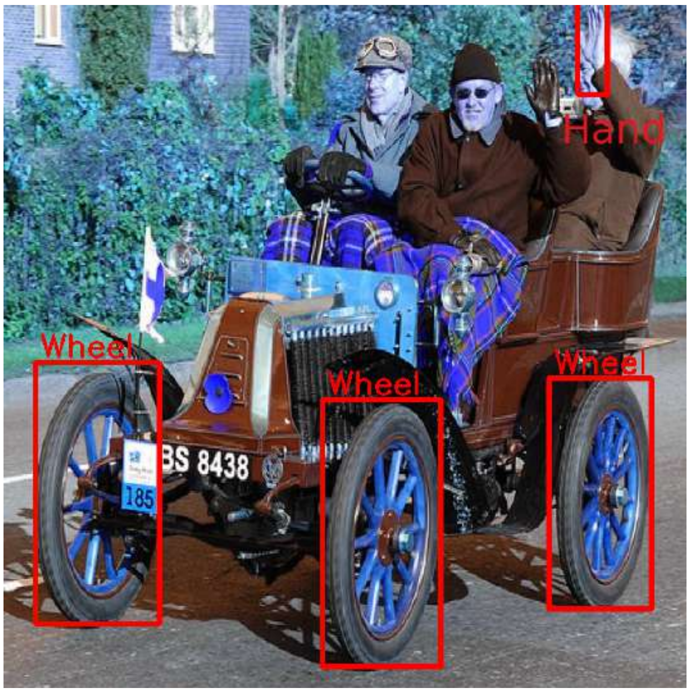


Figure 5.9: Error Case MO example 2. The object parts “Wheel” and “Hand” are detected with high confidence, but cannot be related with a whole object.

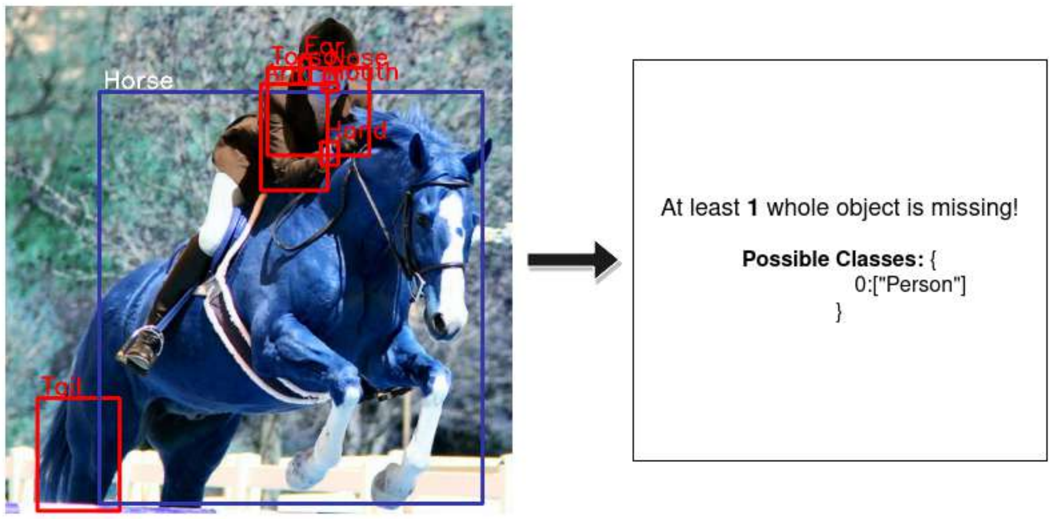
achieves this by leveraging information within the framework's knowledge domain, as well as information from the object part that triggered the error.

Once these missing classes are identified, the ruleset categorizes them into groups, ensuring that every class within a group shares common parts with the other classes in that group, in accordance with the knowledge domain. Finally, the ruleset provides a count of the missing whole objects, which corresponds to the number of groups formed, along with a list of potential classes for the missing objects. The aforementioned ruleset is presented in Appendix A.4.1.

The presented handling procedure had the following outputs for the examples shown in Figures 5.8 and 5.9.

The framework accurately identifies that at least one whole object is absent as shown in Figure 5.10, and suggests that it belongs to the class “Person”. Only a single possible class is presented because most of the detected parts are unique to the class “Person” (e.g., “Hand”, “Mouth”), according to our knowledge repository. In such instances, our ruleset is designed to handle this information appropriately.

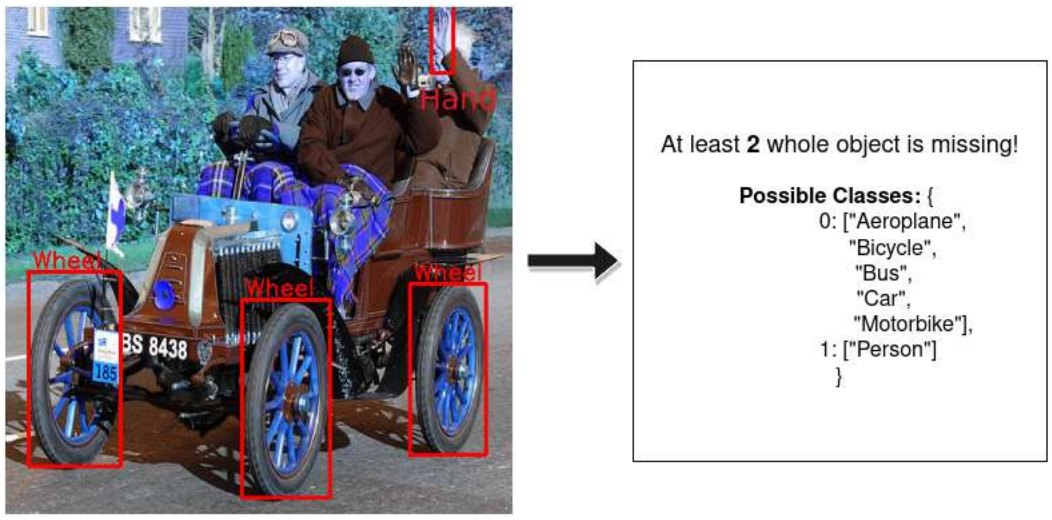
In the case displayed in Figure 5.11, in the outputs is being reported that at least two whole objects were missed by the object detector. This results from the fact that within our knowledge domain, there are no classes that share both the parts “Wheel” and “Hand”. Consequently, two sets of potential classes are presented for each missing object.



At least **1** whole object is missing!

Possible Classes: {
0:["Person"]
}

Figure 5.10: Error Case MO handling example 1. The framework reports that at least 1 whole object with label “Person” should have been returned by the object detector.



At least **2** whole object is missing!

Possible Classes: {
0: ["Aeroplane",
"Bicycle",
"Bus",
"Car",
"Motorbike"],
1: ["Person"]
}

Figure 5.11: Error Case MO handling example 2. The framework reports that at least 2 whole objects are missing. The possible classes for one of the undetected whole objects include “Aeroplane”, “Bicycle”, etc. The only possible label for the other missing object is “Person”.

5.3.5 Wrong detection coordinates

As we explored various error scenarios, it became evident that some errors were originating from inaccurate coordinates provided by the object detection model for the bounding boxes of detected objects. An illustrative example of this issue is presented in Figure 5.12 (where we focus exclusively on the relevant errors while omitting the remaining detections). In this example, the object detector successfully identifies and classifies the object "Person", along with the object parts "Hand" and "Foot", all with high confidence scores.

As outlined in Section 4.3.3, our approach incorporates an inclusion ratio of 75% to discover spatial associations between whole objects and their constituent parts. Apparently, this inclusion ratio cannot be satisfied for the detections within the presented image example. Consequently, our error discovery ruleset erroneously reports two Case MO errors. This misclassification could lead to inaccurate assumptions and suboptimal framework performance.

To address this issue, we implement an initial filtering step for detected errors, as illustrated in Figure 5.13. By substantially reducing the inclusion ratio to a negligible value and then re-executing the framework, we are able to separate from the initially discovered errors, the inaccurate coordinates errors. In the case of coordinate inaccuracies, the framework issues a warning about the affected bounding boxes.

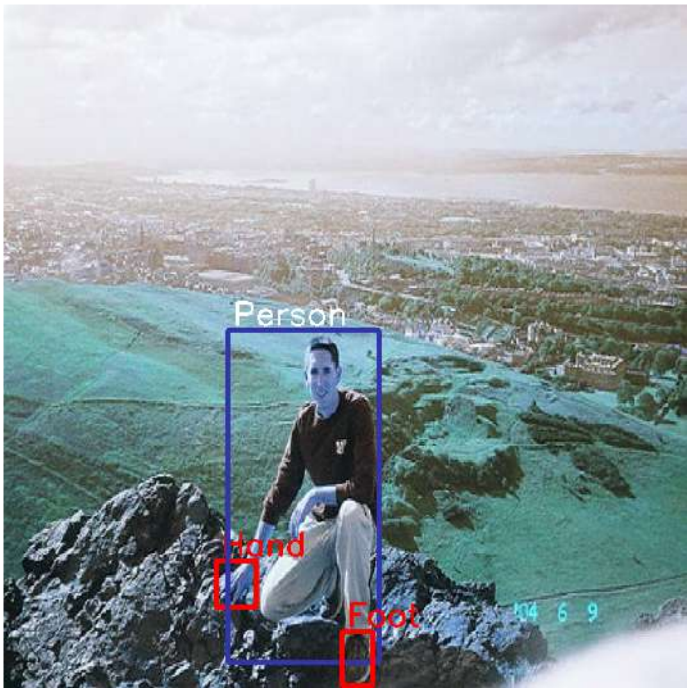


Figure 5.12: Example image where errors are detected due to inaccurate bounding boxes coordinates. The bounding box of the "Person", should be larger to include the object parts "Hand" and "Foot".

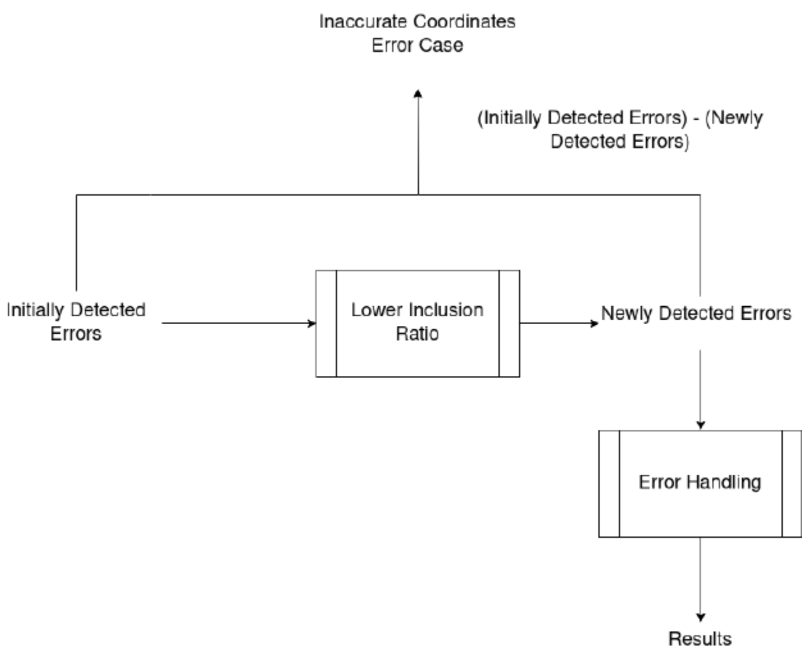

Inaccurate Coordinates Error Case
(Initially Detected Errors) - (Newly Detected Errors)
Initially Detected Errors
Lower Inclusion Ratio
Newly Detected Errors
Error Handling
Results

Figure 5.13: Errors Filtering Pipeline.

5.3.6 Interesting Cases

Observing the outputs of the presented methodology of identifying errors in the outputs of the object detection model, some interesting cases showed up. Following, some examples cases will be presented to showcase how the deployed model, the knowledge domain used or even the annotations of the ground truth have an effect in our method.

Figure 5.14 illustrates error case P, where the object detector identified a “Sheep” (indicated by the blue bounding box) and a “Nose” (indicated by the yellow bounding box) in the image. Notably, the “Nose” detection received a low confidence score. What makes this case intriguing is that it’s entirely logical for a sheep to have a nose, aligning with commonsense. However, in the Semantic Pascal-Part Ontology that we use as ground truth, such connection doesn’t exist i.e., there is no $meronym$(nose,sheep). As a result, in the ground truth annotations regarding the objects present in an image, there is no object part classified as “Nose”. Consequently, the error detection was valid based on the object detector’s output and the ontology.

It’s worth noting that this error was identified solely by leveraging the ontology as our knowledge domain. In contrast, when we employ our methodology to construct the knowledge domain using ConceptNet, naturally, an association between “Nose” and “Sheep” emerges due to common knowledge. In this scenario, the error may go undetected because, according to the knowledge domain, such an association is consistent with commonsense.

In some other scenarios that may arise, we encounter situations similar to the one depicted in Figure 5.15. In this particular case, our framework incorrectly identified an error case P. In the image the object detection model detects an object part "Body" with low confidence. Due to its coordinates it can be spatially associated with the whole object "Person". The two cannot be semantically associated because they belong to different contexts according to their label category (Person, Artifact). Hence, the conditions have been met for the framework to arise an error case P. However, in the ground truth annotation there is no annotated object regarding the yellow bounding box. Clearly, there is an object within the image, and the object detector successfully identifies it. However, since it does not align with the ground truth, the error is categorized as a False Positive.

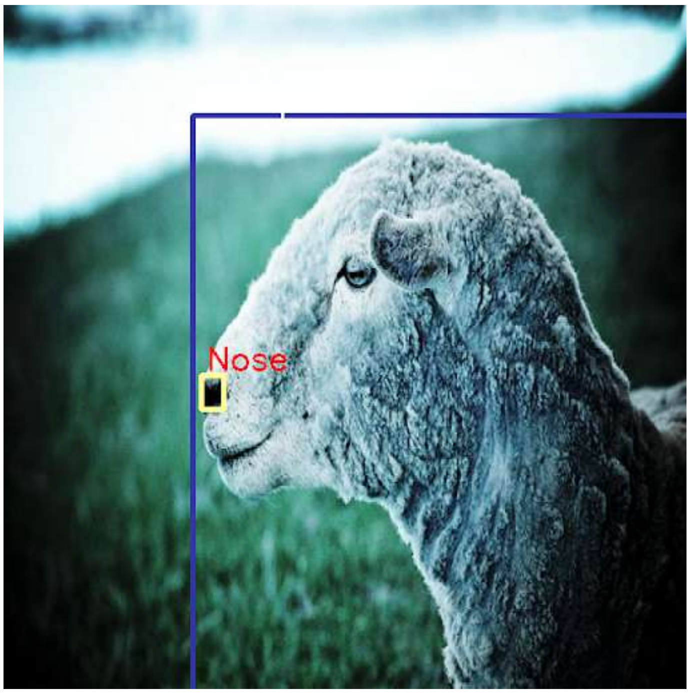


Figure 5.14: A low-confidence (yellow) bounding box "Nose" is detected that is spatially associated with a detected whole object "Sheep". An error case P is triggered, because in the ground truth there is no $meronym$(nose,sheep), hence the yellow box was misclassified. However, a sheep having a nose is not against commonsense knowledge. Using the knowledge domain constructed via ConceptNet, this error cannot be detected.

In Figure 5.16, an illustrative case highlights the impact of confidence scores on the suggested approach for handling detections. Initially, in the image on the left were detected a whole object "Bicycle" and an object part "Leg" that can be spatially associated. Both of these detections fall below the confidence threshold, and there's no semantic connection between them, leading to the accurate identification of an error case, referred to as PO.

However, when our framework attempts to rectify this error, a peculiar situation arises. This is because the "Leg" receives a higher confidence score than the "Bicycle", causing

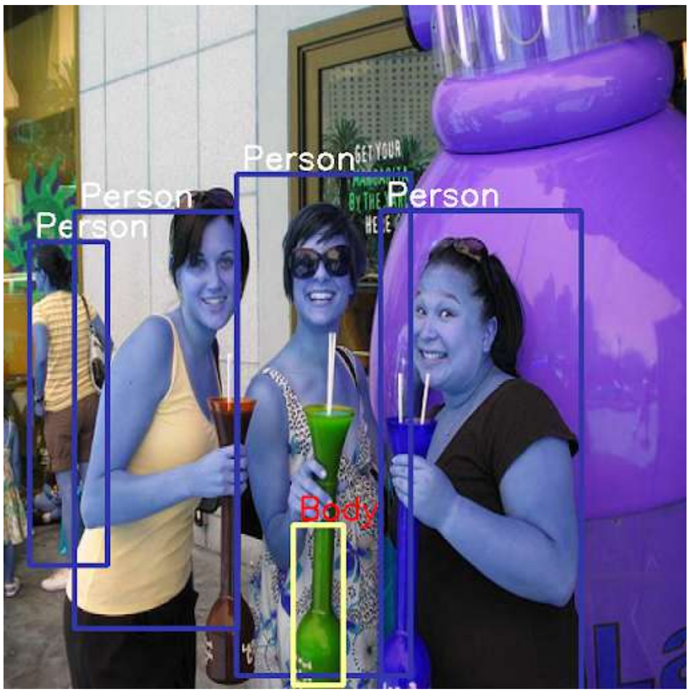


Figure 5.15: An object part “Body” (yellow box) is detected by the object detection model and causes a case P error. However, no object exists in the ground truth annotations regarding the yellow bounding box. Hence, this error detection case was incorrect.

the framework to relabel the latter as “Person” (as seen in the right image). Clearly, the incorrect detection pertains to the “Leg”, and due to the confidence scores provided by the model, the framework fails to resolve the error correctly.

Interestingly, in the initial detection, no “Person” was identified by the model. Nevertheless, in the ground truth annotations, a “Person” does indeed exist. Furthermore, the “Bicycle” also exists. One might describe this handling outcome as “neutral” since the “Bicycle” is erroneously discarded, but a “Person” is accurately classified.

Depicted in Figure 5.17, we encounter a case where our error-handling process leads to incorrect label changes and a decline in object detection performance. The “Wheel” object part (indicated by the yellow box) is detected with a confidence score below the designated threshold in the left image. Furthermore, the “Wheel” shares a spatial relationship with a detected “Person”, although no semantic validation exists. Surprisingly, the object detection model fails to identify the existing “Bicycle” in the image, resulting in our methodology erroneously triggering an error case, denoted as P.

In an attempt to address this error, the handling process mistakenly relabels “Wheel” to “Chair”, as illustrated in the right image, leading to a decrease in overall object detection performance.

Lastly, another interesting insight that was provided observing the outputs of our

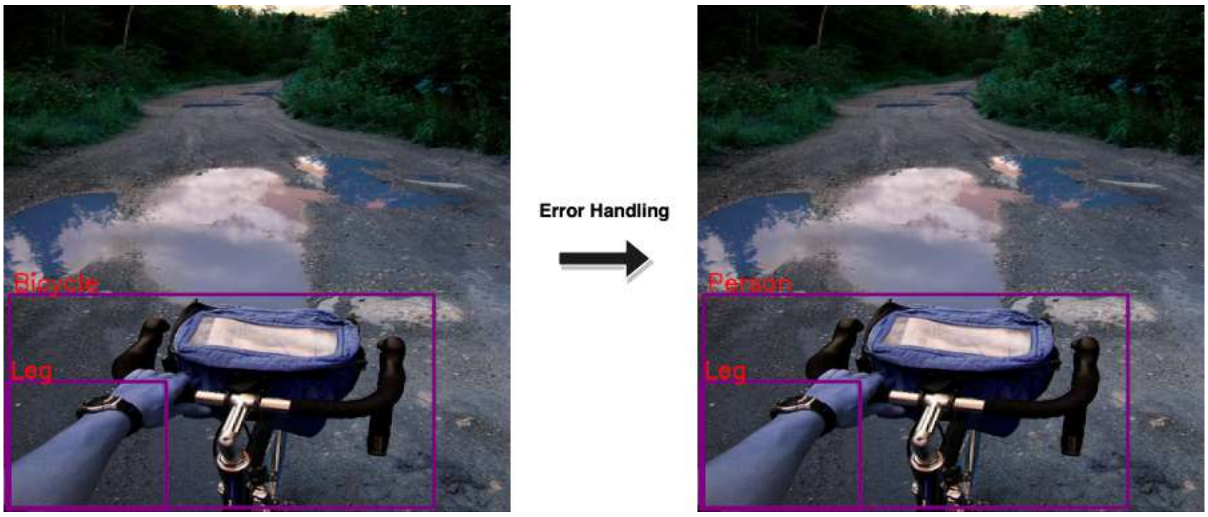


Figure 5.16: Both a "Bicycle" and a "Leg" are detected with confidence scores below the threshold in the image (left), leading to the correct identification of an error case PO due to their lack of semantic connection. However, when the framework attempts to rectify the error (right), the fact that "Leg" is assigned a higher confidence score than "Bicycle" results in the relabeling of the "Bicycle" as "Person". This handling outcome can be described as "neutral", as the "Bicycle" is wrongly dropped, but a "Person" is correctly classified.

methodology is depicted in Figure 5.18. The object detector detects an object part classified as "Headlight" with high confidence score, within the image. However, the object box cannot be spatially related with any whole object in the image. Driven by that information, the framework reports an error MO case, i.e. that a whole object is missing from the outputs generated by the object detection model. Moreover, the error handling procedure suggests that the missing whole object belongs to one of the dataset's vehicle classes (such as bus, car, bicycle, etc.). It is evident that a car is present in the image but was not detected. Nonetheless, there are no corresponding ground truth annotations for a car or any of its components, which leads to the conclusion that the error detection is incorrect, but clearly is not. Examining such error cases can provide valuable insights into potential inaccuracies or missing annotations within the dataset.

5.4 Summary

In this chapter we explained how we expanded our framework to identify possible errors in the object detection model outputs, i.e., the $hasObject$ tuples. Deploying the **Error Detection Ruleset** enabled the discovery and categorization of errors, while utilizing the **Error Handling Module**, our framework deals with the detected errors according to their type. Overall, our methodology reports very promising results as shown in Chapter 7. In

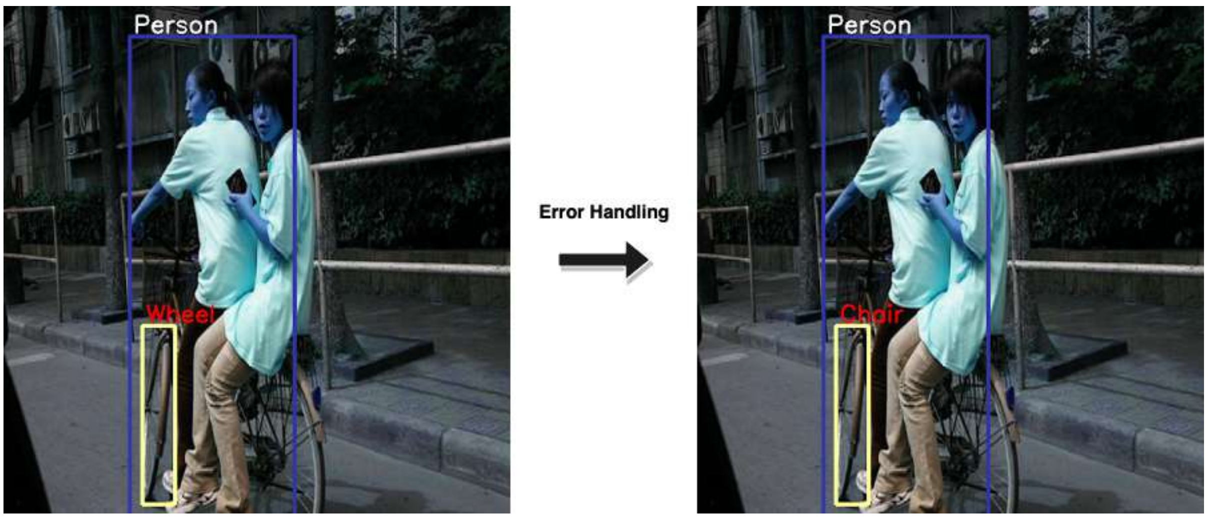


Figure 5.17: Figure showcases a scenario where our error-handling process results in an incorrect label change. Specifically, a “Wheel” object part (indicated by the yellow box) is detected with a confidence score below the specified threshold in the left image. This “Wheel” is spatially connected to a detected “Person”, though no semantic validation exists. The object detection model fails to recognize the existing “Bicycle” in the image, leading our methodology to trigger an erroneous error case P. In an attempt to rectify this error, the handling process inaccurately relabels the “Wheel” as “Chair”, as depicted in the right image, ultimately decreasing the overall object detection performance.

particular in some cases the precision score of this method was equal to 100%. Nevertheless, the framework’s ability to detect possible errors in the outputs of an object detection model can be further enhanced. A possible upgrading is the integration of additional information sources about the objects in an image, and tools for more robust and precise results. Furthermore, the error handling techniques used in this work, did not met our expectations, i.e., improving the task of object detection. Hence, a couple of more refined techniques are proven necessary.

However, the presented methodology provides the first step in quality control of an object detection model, enabling subsequent processes to thoroughly examine and rectify the identified errors.

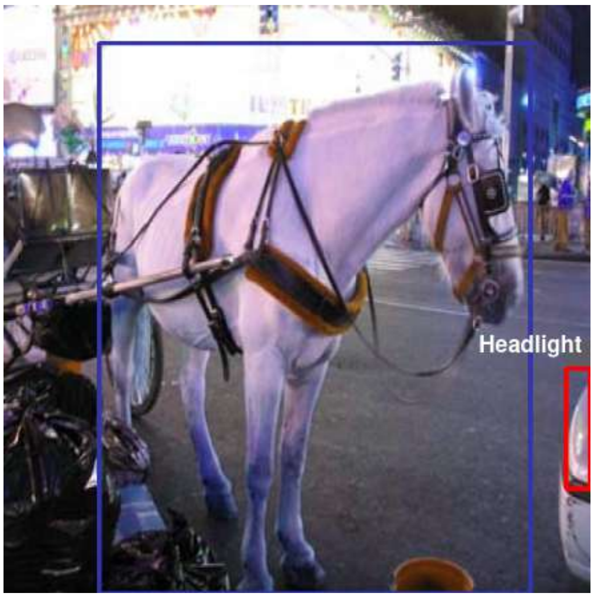


Figure 5.18: Object detector identifies a "Headlight" (red box) with high confidence, but does not relate it to a whole object, triggering an MO error case. The missing object is likely a vehicle (e.g., car), but such object is not annotated in the dataset. Examining such cases may reveal incorrect or missing annotations in the dataset.

Chapter 6

Implementation

In this chapter, we provide an in-depth description of the execution process of our framework, named *PReDeCK*. *PReDeCK's* execution pipeline is presented visually in Figure 6.1. Initially, the framework exports all the available data about the labels of the dataset from ConceptNet. The data is retrieved by sending HTTP queries to ConceptNet through python for every label. Subsequently, this data undergoes the filtering process, and subgraphs for each label are created and stored locally in Neo4j. This process is repeated for the derived concepts as well, to export information about the neighboring nodes of the dataset's labels that are available in ConceptNet. Consequently, the information that the subgraphs hold is represented in ASP format. Once this process is concluded, the generated facts are integrated into NeurASP.

The ***meronym*** **Enrichment ASP Ruleset** leverages the previously generated facts to inference additional $meronym$ relations among the labels within the dataset. Its objective is to attain the utmost coverage of these relations that exist in the ground truth. After the execution of the enrichment ruleset, *PReDeCK's* knowledge domain is complete. To enhance efficiency, this process was conducted external to the framework. Instead, the knowledge domain was inferred, materialized, and then incorporated into the framework as facts.

Once the knowledge domain is fully established, we initiate the process by inputting the Test Set's images into *PReDeCK*. For each image, the object detector outputs all the detected objects, which are also represented in ASP within NeurASP. Following that, the ***partOf*** **Discovery ASP Ruleset** comes into play, utilizing both the knowledge domain and the detector's outputs to infer $partOf$ relations among the identified objects within the image.

The **Error Detection Ruleset** leverages all available information to identify potential errors in the object detector's outputs. These detected errors are stored locally in a JSON file. In the event that any errors are discovered within an image, this information is promptly forwarded to the **Error Handling Module**. The Error Handling Module is tasked with the responsibility of addressing these identified errors according to their type, and, if necessary, making modifications to the data. Following any such modifications,

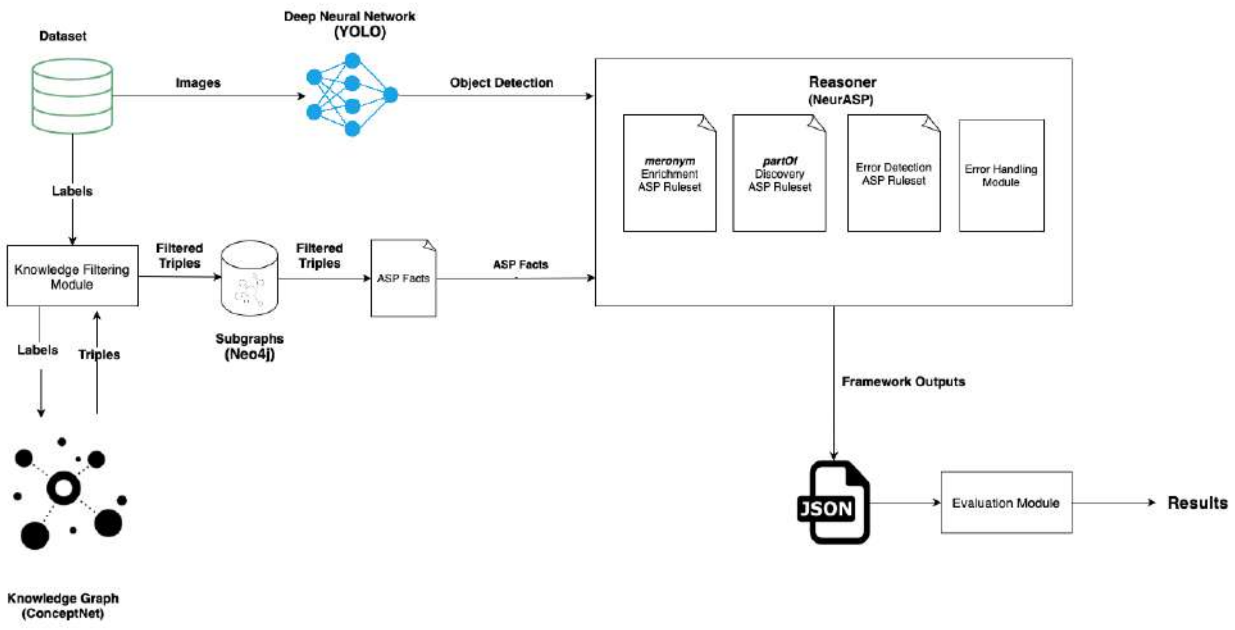


Figure 6.1: *PReDeCK*'s Execution Pipeline

PReDeCK performs a re-evaluation to check for any newly introduced errors. This iterative process is imperative, as resolving certain error cases may trigger the emergence of other error cases. Ultimately, when all errors have been effectively addressed, the results are stored in a separate local JSON file.

Finally, the **Evaluation Module** uses the framework's outputs to assess its effectiveness on multiple fronts.This includes evaluating its performance in discovering $partOf$ relations between objects within an image, its ability to pinpoint erroneous outputs from the object detection model, and its accuracy in resolving them.

It's of note that our methodology and implementation have been designed with a high degree of generalization. With relatively minor adjustments and a modest amount of effort, the framework can be readily adapted to a wide range of tasks and diverse datasets.

6.1 Framework set-up

Instruction on how to set the framework up and the necessary tools are presented below.

NeurASP Framework. First, install the NeurASP framework as instructed in `https://github.com/azreasoners/NeurASP/tree/master`. Instead of installing the packages that are listed in the previous repository, install the provided anaconda environment running the following commands:

```
conda env create -n predeck --file env.yml
conda activate predeck
```

Then, replace the files *neurasp.py* and *mvpp.py* with the ones provided in the submitted repository. In *"NeurASP"* folder, insert the file *string_parser.py* and the folder *"ConceptNet"*. Moreover, insert the folder *"PReDeCK"* in *"NeurASP/examples/"*.

Object detection model and Dataset. While in the working directory *"NeurASP/examples/PReDeCK"* the following steps must be done. Install YOLOv5 according to the instructions found in `https://github.com/ultralytics/yolov5`, and also download the Semantic Pascal-Part dataset we used in our work, from `https://universe.roboflow.com/pascalpart/pascal-part-fquij`. When the download is complete, copy the images of the test set in a folder in the current directory.

Running the framework. The provided folder, *"PReDeCK"*, includes a configuration file, named *config.yaml*. In the configuration file one can fill the required fields, accordingly. These fields include the specification of the images' directory path as well their size (in the provided dataset the size is 640). The model that is to be deployed for the execution must be specified in the file as well, along with path of the output file where the results will be recorded. Finally, the user can specify which of the provided experiments/methods to be executed and enable or disable the evaluation process. When all the aforementioned fields are filled successfully, the framework can be executed using the command:

```
python run.py
```

Commonsense Knowledge exportation via ConceptNet. For the given methods, the knowledge domain is already incorporated in the python scripts. If the user wishes to construct a different domain from ConceptNet, or enhance the existing, all the necessary files are provided. First, you have to initialize a local graph through *Neo4j*, and insert the credentials in the python files, when needed. Next, by running `python neo_concept_categories.py` the local graph that includes the requested information is ready. Note that all the queried labels and relations can be modified inline in the given scripts. At last, to represent the knowledge derived from ConceptNet in ASP, run `python create_CNrules_file.py`. To avoid, the previous steps, we have provided the facts represented in ASP in the file *"CNrules.txt"*.

In general, the method is quite adaptable and can be configured according to the needs of the user.

ASP Rulesets. All the ASP rules formulated for this study, can be found in the *"ASP_PReDeCK Rules.ipynb"* notebook.

Object Detection Evaluation. The evaluation for the object detection models was carried out using the following tool `https://github.com/rafaelpadilla/review_object_detection_metrics`.

The instructions stated above along with all the necessary sources and files can be found in the following github repository: `https://github.com/Kmrs97/PReDeCK/tree/main`

Chapter 7

Evaluation

In this chapter, we describe the evaluation process for the presented methodologies, we report the results, and discuss strengths and limitations.

7.1 Setting

The evaluation was conducted using a test set sampled from the Pascal-Part Dataset, maintaining the same class proportions as the original dataset. This test set comprises 991 images, covering all the original dataset's classes. The annotations for these images serve as the ground truth against which we compare our results. In total, the test set includes 24,512 objects, 154,558 parts, and 15,811 "part of" ($meronym$) relations.

For object detection, we employed a YOLOv5l model, fine-tuned in three different ways as described in Section 4.3.2.

The commonsense knowledge integrated into the framework for the experiments was constructed through various means. The knowledge domain is either created from various adaptations of information extracted from ConceptNet (including variations like no inference rules or incorrect information) or by considering the Semantic Pascal-Part Ontology provided alongside the dataset.

7.2 Design of Experiments

The Design of Experiments section provides information about the hypotheses that were made within the evaluation process of the proposed framework. Additionally, the evaluation metrics deployed for the assessment of our approach, are explained in detail.

7.2.1 Evaluation Measures

The evaluation of $partOf$ relation and errors detection relies on the object detection task. For that reason, we start this section by describing which detected objects were considered correct and which ones were discarded. Then, we proceed with the evaluation measures for $partOf$ relation detection between objects. Finally, we elaborate on how we assess the errors by the object detector, that were identified by the framework.

Object detection. To evaluate the performance of the fine-tuned YOLO model that we used for object detection, we employ *mean Average Precision (mAP)*. Average Precision (AP) is calculated for each class across all images, and mAP is defined as the mean of those AP scores. To compute AP, the predicted bounding boxes are compared with the ground truth annotations, in terms of localization and classification.

For localization, we use the *Intersection over Union (IoU)* value shown in (7.1), which is equal to the area of the intersection between the predicted and ground truth bounding boxes, divided by the area of their union.

$$\text{IoU} = \frac{\text{Area of Intersection}}{\text{Area of Union}} = \frac{A \cap B}{A \cup B} \tag{7.1}$$

If the IoU value between a predicted bounding box b and a ground truth bounding box b' exceeds a specified threshold (usually 0.5), and the predicted label of b is the same as the label of b', then the object prediction is considered correct; otherwise the object prediction is incorrect.

PartOf detection. A $partOf(b1, l1, b2, l2)$ relation suggestion is considered as True Positive (TP), if it satisfies the conditions below:

- Both objects are detected correctly, i.e., $b1$ (with label $l1$) and $b2$ (with label $l2$) are correctly matched with bounding boxes $b1'$ (with label $l1'$) and $b2'$ (with label $l2'$), respectively, as given in the ground truth.
- There is a $partOf$ relation in the ground truth annotations between the examined objects, i.e., $partOf(b1', l1', b2', l2')$.

Accordingly, a predicted $partOf(b1, l1, b2, l2)$ relation that does not satisfy all the conditions above is considered as a False Positive (FP), while the absence of a $partOf(b1, l1, b2, l2)$ relation prediction for a $partOf(b1', l1', b2', l2')$ relation in the ground truth is considered as a False Negative (FN). Overall, the evaluation results for the $partOf$ relation detection task are reported in terms of precision $P = TP/(TP{+}FP)$, recall $R = TP/(TP{+}FN)$, and $F1 = 2PR/(P + R)$.

Error Detection. Depending on the type of error detected, a True Positive (TP) may vary. In instances of errors categorized as P, O, and PO, denoting errors identified in either an object part or the whole object (or both), a TP occurs when the bounding boxes associated with an error are actually misclassified according to the ground truth. For example, if an error is detected and reported using the predicate $errorCaseP(b1)$, a TP is confirmed if $b1$ was indeed misclassified. In the case of error case PO, where two bounding boxes triggered the error, it suffices for at least one of them to be incorrectly identified for a TP to be counted.

A precise detection of a case MO error is considered, if there are actually undetected whole objects in the associated image. For this particular scenario the positives and negatives are measured in number of images and not errors, because a number of bounding boxes that trigger a case MO error may refer to the same missing whole object.

The Precision metric is employed to assess the detected errors. For Case MO errors, the Recall is also reported.

Error Handling. As described earlier, for cases P,O and PO through the error handling process, the model's output labels may undergo modifications. Consequently, the process is also evaluated with Mean Average Precision (mAP).

However, error case MO is treated differently, as we not only detect the absence of whole objects in the detections, but also strive to ascertain the quantity of undetected objects and their potential labels. Therefore, the framework's precision in achieving this objective requires evaluation, for which we have developed four distinct evaluation approaches.

1. Firstly, as mentioned earlier, we verify whether undetected whole objects error is correctly reported (MO1).
2. Secondly, we employ a variation of the first method and check if the framework accurately identifies the exact number of missing whole objects, too (MO2).
3. Thirdly, we validate whether the reported possible classes align with the actual classes of the missing objects (MO3).
4. Lastly, we utilize a combination of the second and third methods for a more rigorous assessment (MO4).

Employing these multiple approaches enables a comprehensive assessment of the framework's capability to identify the absence of whole objects, determine the number of undetected whole objects, and infer their potential labels. This multifaceted evaluation strategy allows for a more nuanced and robust understanding of the framework's performance in handling error case MO.

7.2.2 Hypotheses

***PartOf* Detection Methodology.** Our approach to discover $partOf$ relations between the detected objects in an image, named *PReDeCK*, was compared with (i) a *visual baseline* approach that considers only visual features for the detection of the $partOf$ relations, (ii) *PReDeCK*$_{GT}$ that uses the information available in the Semantic Pascal-Part ontology (i.e., the ground truth) and therefore exhibits ideal performance, (iii) *PReDeCK*$_{CN}$, a variation of our suggested methodology that utilizes knowledge from ConceptNet, but omits the inference phase and (iv) *PReDeCK*$_{Noisy}$, where the knowledge domain was intentionally imputed with false $meronym$ information while ignoring correct ones. We introduced this variation to underscore the significance of having a robust and accurate knowledge domain when addressing these types of problems.

In our evaluation process, we considered the aforementioned comparisons based on a number of hypotheses stated below:

- **Hypothesis 1:** Our approach achieves improved performance in terms of Precision, compared to the *visual baseline*. This hypothesis is grounded on the assumption that the use of both visual and commonsense information can eliminate false $partOf$ outputs.

The *visual baseline* approach that was employed for comparison, follows a naïve strategy to discover the $partOf$ relations, using only the geometric data of the detected objects. If the *inclusion ratio* $ir(b1, b2)$ between two bounding boxes $b1$ and $b2$ exceeds a given threshold θ, then the $partOf$ relation stands:

$$partOf(b1, l1, b2, l2) = \begin{cases} true, & \text{if } ir(b1, b2) \geq \theta \\ false, & \text{otherwise} \end{cases} \text{, where}$$

$$ir(b1, b2) = \frac{area(b1 \cap b2)}{area(b1)}.$$

We empirically set the threshold θ value to 0.75. Note that labels $l1$ and $l2$ are completely ignored.

- **Hypothesis 2:** Commonsense knowledge from problem-agnostic KGs, as used in our proposed methodology, can help achieve performance close to (but cannot exceed) the ground truth approach (PReDeCK$_{GT}$), in terms of precision and recall for the prediction of part of relations.

  The *PReDeCK$_{GT}$* method uses the same ruleset as the presented methodology, but instead of relying on the information extracted from ConceptNet as its knowledge domain, it uses the information obtained from the Semantic Pascal-Part ontology. This information is error-free and contains all possible $meronym$ relations that are present in the dataset's annotations.

  To highlight the significance of the knowledge domain and see how close we get with the knowledge that we extract from ConceptNet to a baseline with complete knowledge, we define two variations for *PReDeCK$_{GT}$*. The first one, *PReDeCK$_{GT1}$*, considers only the type of the predicted label (*Object/Part)* to infer $partOf$ relations. In other words, it uses a variation of the rule presented in Section 4.3.3, where it omits the last predicate of the rule that checks if there is a $meronym$ relation in the knowledge repository. In this case, when two bounding boxes that meet the spatial conditions, are classified as belonging to different types, this is sufficient evidence for deriving a $partOf$ relation among them.

  The second approach, *PReDeCK$_{GT2}$*, uses the rule defined in Section 4.3.3 as is, and all the available information about $meronym$ knowledge, present in the ground truth, to discover the $partOf$ relations between the detections.

- **Hypothesis 3:** PReDeCK's results outperform identically implemented frameworks that lack or rely on false knowledge.

  We compare our method against two variations that utilize smaller parts of our constructed knowledge domain. *PReDeCK$_{CN1}$* uses the labels' types and categories along with information that is directly accessible through the edge "PartOf" of ConceptNet, with no further knowledge enrichment.

PReDeCK$_{CN2}$ discovers the $partOf$ relations without the specification of the type and category for the labels. In this case the following variation of the $partOf$/4 rule is used:

$$\begin{aligned}
&partOf(B1, L1, B2, L2) :- \\
&\qquad candidatePartOf(B1, B2), \\
&\qquad label(\_, \_, B1, L1), \\
&\qquad label(\_, \_, B2, L2), \\
&\qquad meronym(L1, L2, \_).
\end{aligned}$$

Finally, in order to demonstrate the impact of inaccurate or missing information within a knowledge domain on the method's performance, we deploy *PReDeCK*$_{Noisy}$. More specifically, we initially manually linked all the $meronym$ information related to the label "Person" with the labels "Car" and "Bicycle," and subsequently excluded it from the knowledge domain.

Error Detection. The extension of the framework to recognise errors of the object detection model, referred to as *PReDeCK+ED*, was similarly deployed using two different knowledge domains *PReDeCK+ED*$_{GT}$ and *PReDeCK+ED*$_{CN}$, following the presented hypotheses:

- **Hypothesis 1.** Due to the absence of the complete knowledge, *PReDeCK+ED*$_{CN}$ can discover an increased number of errors in the images, but with less precision. On the other hand, we expect *PReDeCK+ED*$_{GT}$ to identify errors with a very high level of precision.

  PReDeCK+ED$_{GT}$, utilizes all the available $meronym$ knowledge that exists in the Semantic Pascal-Part Ontology (i.e. ground truth). In contrast, *PReDeCK+ED*$_{CN}$ relies on knowledge exported from ConceptNet and enriched through the inference phase. Given the importance of precision for optimal performance in this context, we have refined the $meronym$ enrichment ruleset to minimize inaccuracies present in the generated knowledge domain. As a result, fewer $meronym$ relations were inferred, achieving a 72% coverage of the knowledge found in the ground truth.

- **Hypothesis 2.** Deploying a better object detection model can lead to fewer error detections with high precision.

  For the evaluation of the error detection methodology, we deploy all the different variations of the Yolov5 model we fine-tuned, namely PreYoloL,ScrYoloL, and Noisy YoloL.

7.3 Experimental Results

Following, the results of the experiments conducted according to the hypotheses discussed above are presented.

Table 7.1: Object Detection Models Performance.

Model	**mAP (%)**
PreYoloL	54.8
ScrYoloL	41.3
NoisyYoloL	26.2

7.3.1 Object Detection Model

As previously stated within this thesis, it is important to note that the effectiveness of our methodology is directly impacted by the choice of the deployed object detection model. Consequently, we initiate this section by presenting the performance results of the Yolov5 models we have fine-tuned. The Table 7.1 displays the Mean Average Precision (mAP) for each fine-tuned model. These models include:

- PreYoloL, the model was fine-tuned on the dataset using pre-trained weights.
- ScrYoloL, the model was trained from scratch
- NoisyYoloL, the model was trained using noise-imputed training annotations. In this case, manual noise was introduced into the annotations, where 20% of each occurrence of whole object classes in the training set had their labels randomly replaced with other whole object labels.

In the case of the $partOf$ relations discovery method, we employed only the pre-trained model. This decision was made to ensure the object detector's optimal performance, as this approach aligns with the requirement for achieving the highest level of accuracy in the method. In contrast, for the error detection method, we employed all available models to observe the variance in the results.

7.3.2 *PartOf* Relations Discovery

7.3.2.1 *PReDeCK-Visual Baseline* (Hypothesis 1)

The *visual baseline* was able to correctly discover $\sim$7k $partOf$ relations, but a very large amount of false detections ($\sim$23k) were reported, too. This led to a precision of 23.5%, as presented in Tables 7.2 and 7.3. On the contrary, *PReDeCK* was able to decrease the number of FPs by $\sim$68%, elevating the precision to 47.3% (+23.8%) without compromising the recall score. The recall was only decreased by 3.6%.

PReDeCK outperformed the *visual baseline* approach and increased the F1 score by $\sim$13.4%. Hence, by incorporating commonsense knowledge to tackle the discussed problem, we were able to double the precision of the results.

7.3.2.2 *PReDeCK-PReDeCK$_{GT}$* (Hypothesis 2)

The *PReDeCK$_{GT1}$* approach outperformed the *visual baseline* by reporting the same true positives (TPs) while decreasing the number of false positives (FPs) by approximately 63.2%. This led to an increased precision of 45.5%, representing a 22% improvement. Using the information regarding the $meronym$ annotations, *PReDeCK$_{GT2}$* further reduced the number of FPs, yielding a precision equal to 47.6%.

Recall remains unchanged throughout the first three methods because, despite each method using different conditions to establish a $partOf$ relation, the detected boxes remain the same. The recall score does not take into account the false positives, but solely rely on true positives. As previously noted, a true positive (TP) is a prediction that correctly associates two detected objects with a $partOf$ relation according to the ground truth, regardless the inference method. The varying conditions in these methods are designed to eliminate false positives consulting their knowledge domain, while keeping the number of true positives unaffected, thus maintaining consistent recall values.

The results of *PReDeCK$_{GT2}$* can be considered as the best possible results that the suggested methodology has to offer, because it employs the ground truth information about the $meronym$ relations as its knowledge domain. As observed in Table 7.3, the performance of *PReDeCK* is very close to the one reported by *PReDeCK$_{GT2}$*. The precision only slightly decreased by 0.3%, while the recall decreased by 3.6%. *PReDeCK* was able to report scores very close to the highest possible scores because it employs a very broad coverage of $meronym$ relations in its knowledge domain. This was accomplished due the knowledge enrichment ruleset we devised, that was able to achieve a 85% coverage of the ground truth through inference, missing only 15 relations compared to *PReDeCK$_{GT2}$*.

7.3.2.3 *PReDeCK - PReDeCK$_{CN\ variations}$* (Hypothesis 3)

Owing to the lack of information, *PReDeCK$_{CN1}$* achieved a mere 1% recall, as evidenced in Table 7.3. In the knowledge domain of *PReDeCK$_{CN1}$* exist only 4 out of the 100 $meronym$ relations that are present in the ground truth, because is constructed using only the “Part Of” edge that is available in ConceptNet. In this case we explicitly defined that to report a $partOf$ relation between two detected objects, their labels must be associated with a $meronym$ relation within the knowledge domain. So, naturally, it discards most of the $partOf$ predictions, because they fail to meet the conditions of the employed rules. In contrast, leveraging a wider range of the available knowledge that exists in ConceptNet, and enhancing it through our knowledge enrichment rules, *PReDeCK* attained a precision of 47.3%, marking a substantial 40.1% increase in recall, resulting in an overall score of 41.7%. These results highlight the need of enriching the knowledge that one can get from open repositories through reasoning, particularly when the information sources are not specifically tailored to address the target problem.

When omitting information regarding label types and categories, *PReDeCK$_{CN2}$* demonstrated a precision of 46.5%, only marginally lower by 0.8% compared to that of *PReDeCK*. This observation aligns with the fact that providing the framework with the information of the label type (Object/Part) is not a necessity. While data-driven approaches could be highly impacted by the absence of this information, our framework's performance was insignificantly affected. This relies on the fact that in our work we utilize commonsense knowledge, and this information can be implicitly derived through the facts that exist in the constructed knowledge domain.

The introduction of noisy information into the knowledge domain significantly reduced recall to 14.7%, a 27% decrease from *PReDeCK*'s recall performance. In this particular variation, we intentionally introduced incorrect $meronym$ associations into our knowledge repository, linking parts of a person with cars and bicycles (e.g., $meronym(head, bicycle, artifact)$). This decision was motivated by the frequent overlap between these objects in the dataset images, where parts of a person are often enclosed by the detected bounding boxes of cars or bicycles. Consequently, numerous false $partOf$ relations were reported between the parts of a person and the whole objects "Car" or "Bicycle". For instance, in the example shown in 1.1 (top), this approach would erroneously indicate a $partOf(hand, bicycle, artifact)$. Additionally, we deliberately omitted all correct $meronym$ relations associated with the label "Person" to assess the impact on the results. Since the "Person" label is the most frequent in the dataset, discarding all predictions related to "Person" significantly affected recall.

In summary, *PReDeCK* outperformed identical approaches employing incomplete or false information within their knowledge domain.

7.3.2.4 Alternating IoU Value

Our work heavily relies on the object detection task, however our focus is not to strictly detect the objects within an image, instead we want to be able to predict $partOf$ relations amongs them. Observing the results of the object detector, we realised that in many cases, am accurate $partOf$ prediction was discarded due to the IoU value, that is commonly defined as 0.5. Following that observation we decided to decline a little from the usual value and see how this affects the results of the framework. After experimenting with a range of different IoU values for the matching of whole objects and object parts respectively, we concluded that lowering the IoU value for the labels categorized as *Parts*, led to better results in terms of Average Precision. This may be due to the difficulty that an object detection may exhibit in localizing the small area of the parts. In addition, the parts may appear in different shapes and forms in the dataset, adding an additional struggle for the model. For example, the label "Leg", appears in 7 different forms (all the animals and the person have legs).

The results shown in Table 7.4 correspond to an IoU value of 0.3 for matching the object parts. The choice of 0.3 over other alternatives was deliberate, as it enhances the performance of the $partOf$ methodology while simultaneously preserving the integrity of the object detection task. We observe an increase of up to 5.7% in precision and 5.3% in recall, compared to the results of Table 7.3, with an IoU value of 0.5.

Table 7.2: Total True positives (TP), False positives (FP) and False negatives (FN).

Method	**TP**	**FP**	**FN**
Visual baseline	7,157	23,255	8,654
PReDeCK$_{\mathrm{GT1}}$	7,175	8,556	8,654
PReDeCK$_{\mathrm{GT2}}$	7,175	7,872	8,654
PReDeCK$_{\mathrm{CN1}}$	162	258	15,649
PReDeCK$_{\mathrm{CN2}}$	6,600	7,589	9,211
PReDeCK$_{\mathrm{Noisy}}$	2,325	2,264	13,486
PReDeCK	6,600	7,350	9,211

Table 7.3: Precision (P), recall (R) and F1 with IoU = 0.5.

Method	**P (%)**	**R (%)**	**F1 score (%)**
Visual baseline	23.5	45.3	30.95
PReDeCK$_{\mathrm{GT1}}$	45.5	45.3	45.4
PReDeCK$_{\mathrm{GT2}}$	47.6	45.3	**46.42**
PReDeCK$_{\mathrm{CN1}}$	38.6	1	1.95
PReDeCK$_{\mathrm{CN2}}$	46.5	41.7	43.97
PReDeCK$_{\mathrm{Noisy}}$	50.7	14.7	22.79
PReDeCK	47.3	41.7	44.32

Table 7.4: Precision (P), recall (R) and F1 with IoU = 0.3.

Method	**P (%)**	**R (%)**	**F1 score (%)**
Visual baseline	26.3	50.6	34.61
PReDeCK$_{\mathrm{GT1}}$	50.9	50.6	50.75
PReDeCK$_{\mathrm{GT2}}$	53.2	50.6	**51.87**
PReDeCK$_{\mathrm{CN2}}$	52.1	44.8	49.31
PReDeCK	53	46.8	49.71

7.3.3 Error Detection

7.3.3.1 *PReDeCK+ED$_{CN}$* - *PReDeCK+ED$_{GT}$* (Hypothesis 1)

The precision scores for error detection in different cases are presented in Tables 7.5, 7.6, 7.7, and 7.8, corresponding to the four cases discussed in Section 4.3.3. In most instances, precision scores exceed 60%, with some reaching 100%. The lowest score, 37.5%, is observed in Table 7.6, which pertains to the model trained from scratch using ConceptNet knowledge. This score is due to the framework's inability to cover certain essential information. This inability lies to the fact that, to avoid noisy information within the knowledge domain, we employed a variation of the knowledge enrichment ruleset that excluded almost all the rules involving synonym information. Some were not entirely excluded, but were augmented with more strict conditions (e.g. $hypernym$), to infer new $meronym$ relations. Unfortunately, this led to a knowledge domain covering only 72% of the ground truth, compared to the 85%, that was previously achieved. Consequently, crucial information such as $meronym(Hair, Person)$, $meronym(Tail, Horse)$, $meronym(Tail, Sheep)$, and others was omitted.

In the same tables, one can observe the low number of errors detected in cases P, O, and PO, which aligns with our expectations. As previously outlined, we implemented numerous condition rules to detect the possible error cases. Reducing the number of conditions for flagging erroneous outputs, leads to unsatisfactory results because it returns a higher number of false positives, lowering the precision of the framework. Moreover, efforts to address these false positive detections, has a significant negative impact on the mAP of the deployed models.

In summary, adopting more conditions to detect possible errors, resulted in high precision scores, which is the desirable outcome when addressing these fine-grained problems. Additionally, utilizing ConceptNet-derived information as the knowledge domain yielded more error detections, translating to an increased number of False Positives, compared to employing the complete ontology knowledge, aligning with our previously discussed hypothesis.

7.3.3.2 Deploying various object detection models (Hypothesis 2)

In cases P and O, we observe that the ScrYoloL model had the worst performance. This happens because of the large variance in confidence scores when detecting the objects. The model detects some objects with high confidence score, while others with lower, resulting in more false errors in the mentioned cases. On the other hand, when we imputed the training set with noise, we concluded that the NoisyYoloL model outputs its detections with a lower confidence score. Therefore, more case PO errors were identified and even though the precision score compared to the other models is lower, it is a descent score of 72.73% (Table 7.7). As expected, the pre-trained model, PreYoloL, detects a lower number of errors with very high precision. This holds for all error cases except error case MO. As presented in Table 7.8, for both knowledge domains, the model exhibits low performance in terms of Precision. This can partly be explained by the model's high confidence in detecting objects and the incomplete knowledge of the ConceptNet-derived knowledge domain. Diving deeper in the outputs, although a number of cases were spotted, where the model detected an object part with high confidence, both the specific part and its corresponding whole object were missing from the ground truth annotations. NoisyYoloL, naturally fails to detect whole objects in many images. *PReDeCK+ED* is able to detect that absence in ∼65% of those images with a nearly perfect precision score of 97.05% and 93.2% for each knowledge domain respectively.

Table 7.5: Error Case P results.

Model	PReDeCK+ED$_{GT}$			PReDeCK+ED$_{CN}$		
	TP	FP	Precision	TP	FP	Precision
ScrYoloL	7	4	64.64	43	21	67.19
PreYoloL	9	1	90	56	23	70.89
NoisyYoloL	7	5	58.33	32	15	68.09

Table 7.6: Error Case O results.

Model	PReDeCK+ED$_{GT}$			PReDeCK+ED$_{CN}$		
	TP	FP	Precision	TP	FP	Precision
ScrYoloL	3	2	60	3	5	37.5
PreYoloL	2	0	100	5	1	83.33
NoisyYoloL	2	0	100	2	0	100

Table 7.7: Error Case PO results.

Model	PReDeCK+ED$_{GT}$			PReDeCK+ED$_{CN}$		
	TP	FP	Precision	TP	FP	Precision
ScrYoloL	7	0	100	7	0	100
PreYoloL	1	0	100	9	0	100
NoisyYoloL	6	0	100	8	3	72.73

Table 7.8: Error Case MO results (MO1).

Model	PReDeCK+ED$_{GT}$					PReDeCK+ED$_{CN}$				
	TP	FP	FN	Precision	Recall	TP	FP	FN	Precision	Recall
ScrYoloL	331	32	305	91.18	52.04	392	95	244	80.49	61.64
PreYoloL	208	39	278	84.21	42.8	293	158	193	64.97	60.29
NoisyYoloL	560	17	305	97.05	64.74	562	41	303	93.2	64.97

7.3.3.3 Error Handling

The number of detected errors for Cases P, O and PO are relatively low compared to the entire test set. Changing the labels to resolve the specific errors, did not have any significant increase or decrease in the mAP score. While some label changes were accurate, in other cases, especially due to incomplete or missing knowledge, labels were updated erroneously. In some other instances, a label modification may corrected a label, but erroneously changed another in the process. After the error handling process regarding the cases P, O and PO, all outputs aligned with the commonsense knowledge provided by the specified knowledge domain.

Regarding the case MO errors, the additional evaluation methods results are presented in Table 7.9. When detecting the exact number of missing whole objects as defined in MO2, using the ontology knowledge led to precision scores up to 50%. Using the knowledge domain derived from ConceptNet, reduced this score to a peak of 32.17%. This occurred due to the presence of facts exported for ConceptNet, that are aligned with commonsense but are not provided by the ontology. This issue led to more commonalities regarding the parts between the dataset's whole object classes, and consequently narrowed down the number of whole object possible categories that corresponds to the number of undetected objects.

Reporting an accurate possible class for the missing objects (MO3) achieved a precision score up to 66.61% and 44.6% using the Ontology knowledge and ConceptNet knowledge, respectively. Finally, identifying the exact number of missing whole objects and the correct possible class, per the definition of MO4, attained a score up to 46.62% with the Ontology knowledge domain deployed, and 19.73% with the ConceptNet knowledge. The latter score, is somewhat low for the same reasons that were explained before concerning the MO2 score.

Table 7.9: Error Case MO additional evaluation methods Precision (P).

Model	PReDeCK+ED$_{GT}$			PReDeCK+ED$_{CN}$		
	MO2	MO3	MO4	MO2	MO3	MO4
ScrYoloL	50.14	64.71	42.7	31.62	44.34	17.86
PreYoloL	48.99	58.8	40.49	28.38	44.6	19.73
NoisyYoloL	53.03	66.61	46.62	32.17	44.51	12.11

Chapter 8

Discussion

Overall, we came down to the following observations. The integration of commonsense knowledge with visual features offers significant benefits for the problems addressed in this study. Despite our extreme assumption that no training data was used to develop a classifier for the detection of $partOf$ relations, our methodology reports notably high scores in detecting $partOf$ relations between objects that exist in an image. This outcome underscores the significance of integrating commonsense knowledge with data-driven approaches. With a $partOf$ classifier in place, a hybrid system has the potential to attain even higher overall performance scores.

Evidently, as shown in 7.3.2.1, it significantly improves the precision with a negligible impact on recall, when it comes to discovering $partOf$ relations between objects detected in an image. Across almost all experiments, the precision score approximately doubled when compared to the *visual baseline*, which considers only spatial information to report the $partOf$ relations.

Our data extraction methodology, using an external Knowledge Graph (ConceptNet) not designed for the $partOf$ discovery task, achieves results that closely resemble the performance having complete $meronym$ knowledge of the ground truth (see 7.3.2.2). Furthermore, our methodology minimizes the inclusion of noisy information to a great extent. The importance of having a robust and accurate knowledge domain, when addressing this kind of problems, is showcased observing the results reported in 7.3.2.3.

In addition, by utilizing the outputs generated by the previous method, our framework was extended to identify false or missing outputs of the object detector. The precision in detecting each case was quite satisfactory, with notable success in identifying instances where the deployed model failed to detect certain whole objects in images (Case MO).

Despite the effort of reducing the inclusion of noisy data, even the slightest number of noisy information in the knowledge domain affects the error detection task. Incomplete or inaccurate knowledge lead to more errors' triggers, that in the majority of the cases, result to a decrease in precision, as shown in 7.3.3.1.

By implementing the framework according to the hypothesis outlined in 7.3.3.2, which defines that deploying high-performance object detection models can yield more precise error detections, we successfully identified inaccuracies and missing annotations in the deployed dataset by observing the outputs.

Moreover, by inspecting the *grounded facts* for both tasks, it is easier to interpret how the method ended up with the specific results, due to the declarative nature of the rulesets.

8.0.1 Limitations

The limitations of our work are the following. First, our methodology heavily relies on the performance of the neural network deployed for the object detection task. Our results are based on fine-tuned models, whose performance is not ideal for the given dataset. This can heavily impact the overall accuracy of the $partOf$ detection methodology. Improving the neural network's performance is essential for enhancing the effectiveness of this methodology. However, the use of a high-accuracy object detector may affect the efficiency of the error detection methodology. In this case a dilemma is presented: while it may lead to more precise error detections, it also risks diminishing the importance of the proposed error detection methodology due to the robustness of its outputs.

Second, although the construction of our knowledge domain was automated, involving the automatic extraction and filtering of information using our method, the determination of the type and category of the labels required manual effort. Before querying ConceptNet with the labels, we conducted a comprehensive search to manually define the category of each label, specifying the context of use, following the sense label attribute, provided by ConceptNet (e.g., emphasizing that we are interested in the "cat" as an animal rather than any other interpretation of the term). Additionally, we manually separated the labels into Objects and Parts based on the information provided along with the dataset.

Despite the effort to minimize noise from CN, a lot was produced affecting performance. When discovering $partOf$ relations, some noisy knowledge does not have high impact, but when detecting errors it can be catastrophic. For this reason, for the latter method, we limited our inference rules, to avoid noisy facts. With the numerous current efforts to improve the quality of commonsense KGs, e.g., the CSKG [23] or the ATOMIC2020 [21] datasets, we expect that this limitation will become less central in the future, at least for the given domain.

Finally, during the inference phase, we noticed a significant amount of time consumed by grounding. To overcome this delay, we opted to generate the facts separately and then insert them into our rule-set. While this workaround helped reduce processing time, it added complexity to the overall process. The time inefficiency issue was somewhat anticipated, given the utilization of state-of-the-art tools in a field currently undergoing intense research, with expectations of more time-efficient approaches emerging in the future. In our work, the extended processing time is due to NeurASP's use of clingo to ground the entire program and enumerate all stable models. The authors in [42] acknowledge that the current implementation of NeurASP is not highly scalable, and the mentioned issues will be addressed in future developments. The plan involves transitioning to approximate inferences instead of exact inferences, which is expected to enhance scalability and efficiency.

8.0.2 Pre-trained Models Association

To showcase the generic nature of our methodology and to explore the possibilities for mitigating the first limitation mentioned above, and the extended benefits that the integration of commonsense knowledge with data-driven methods has to offer, we performed an additional experiment for the object detection task. In this experiment, we used a model that was pre-trained on a different set of images, with different output classes. The model was pre-trained on the COCO dataset [30], that has 80 object classes. Out of these classes, only 16 are syntactically similar with the classes of the Pascal-Part Dataset.

Leveraging the information about the synonyms from ConceptNet, we were able to match 3 more classes. Evaluating the model on our test set, considering only the 19 classes that were matched, we had the following results: in terms of mAP, the model using only exact string matching, had a score of 62.2%. On the other hand, using the synonyms knowledge, there was an increase of 10.7%, resulting to a mAP of 72.9%.

Chapter 9

Conclusions

This paper presents a framework, built upon NeurASP, for discovering $partOf$ relations between objects detected in an image. Moreover, the proposed framework was expanded to identify errors of the object detection model. The results show that incorporating more valid information into a knowledge domain of a system improves the performance. Our framework confirms that combining commonsense knowledge with data-driven approaches, leads to better results, in contrast to using only the latter. Additionally, we demonstrate a generalized methodology for extracting and cleaning data from ConceptNet.

As future work, we would like to expand our rulesets with additional inference capabilities to further increase precision for the specified problems. For instance, formulate cardinality constraints for the $partOf$ task, to filter out incorrect inferences. Furthermore, we would like to incorporate more state-of-the-art techniques for image understanding into the framework and combine the provided information with the current, for more robust outputs. Finally, we plan to enhance the error handling technique to contribute to the object detection task. For example, attempt to draw the bounding boxes of some missing objects, as detected by the framework (Case MO).

Bibliography

[1] Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. Dbpedia: A nucleus for a web of open data. In *Proceedings of the 6th International The Semantic Web and 2nd Asian Conference on Asian Semantic Web Conference*, ISWC'07/ASWC'07, page 722–735, Berlin, Heidelberg, 2007. Springer-Verlag.

[2] Samy Badreddine, Artur d'Avila Garcez, Luciano Serafini, and Michael Spranger. Logic tensor networks. *Artificial Intelligence*, 303:103649, 2022.

[3] Djallel Bouneffouf and Charu C. Aggarwal. Survey on applications of neurosymbolic artificial intelligence. *ArXiv*, abs/2209.12618, 2022.

[4] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PP, 2016.

[5] Xianjie Chen, Roozbeh Mottaghi, Xiaobai Liu, Sanja Fidler, Raquel Urtasun, and Alan Yuille. Detect what you can: Detecting and representing objects using holistic models and body parts. CVPR '14, page 1979–1986, USA, 2014. IEEE Computer Society.

[6] Palash Dey and Sourav Medya. Manipulating node similarity measures in networks. In *AAMAS*, pages 321–329, 2020.

[7] Natalia Díaz-Rodríguez, Alberto Lamas, Jules Sanchez, Gianni Franchi, Ivan Donadello, Siham Tabik, David Filliat, Policarpo Cruz, Rosana Montes, and Francisco Herrera. EXplainable neural-symbolic learning (x-NeSyL) methodology to fuse deep learning representations with expert knowledge graphs: The MonuMAI cultural heritage use case. *Information Fusion*, 79:58–83, 2022.

[8] Ivan Donadello and Luciano Serafini. Integration of numeric and symbolic information for semantic image interpretation. *Intelligenza Artificiale*, 10(1):33–47, 2016.

[9] Ivan Donadello, Luciano Serafini, and Artur S. d'Avila Garcez. Logic tensor networks for semantic image interpretation. In *International Joint Conference on Artificial Intelligence*, 2017.

[10] Glenn Jocher et al. ultralytics/yolov5: v7.0 - YOLOv5 SOTA Realtime Instance Segmentation, 2022.

[11] Mark Everingham, Luc Van Gool, Christopher Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *International Journal of Computer Vision*, 88:303–338, 2010.

[12] Yuan FANG, Kingsley Kuan, Jie Lin, Cheston Tan, and Vijay Chandrasekhar. Object detection meets knowledge graphs. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence*, pages 1661–1667, Melbourne, Australia, August 19-25 2017.

[13] Nadime Francis, Alastair Green, Paolo Guagliardo, Leonid Libkin, Tobias Lindaaker, Victor Marsault, Stefan Plantikow, Mats Rydberg, Petra Selmer, and Andrés Taylor. Cypher: An evolving query language for property graphs. In *Proceedings of the 2018 International Conference on Management of Data*, SIGMOD '18, page 1433–1445, 2018.

[14] Anurag Garg, Niket Tandon, and Aparna S. Varde. Csk-sniffer: Commonsense knowledge for sniffing object detection errors. *CEUR Workshop Proceedings*, 3135, 2022. Publisher Copyright: © 2022 Copyright for this paper by its authors.; 2022 Workshops of the EDBT/ICDT Joint Conference, EDBT/ICDT-WS 2022 ; Conference date: 29-03-2022.

[15] MARTIN GEBSER, ROLAND KAMINSKI, BENJAMIN KAUFMANN, and TORSTEN SCHAUB. Multi-shot asp solving with clingo. *Theory and Practice of Logic Programming*, 19(1):27–82, 2019.

[16] Wandemberg Gibaut, Leonardo Pereira, Fabio Grassiotto, Alexandre Osorio, Eder Gadioli, Amparo Munoz, Sildolfo Gomes, and Claudio dos Santos. Neurosymbolic ai and its taxonomy: a survey. 2023.

[17] Filippos Gouidis, Alexandros Vassiliades, Theodore Patkos, Antonis A. Argyros, Nick Bassiliades, and Dimitris Plexousakis. A review on intelligent object perception methods combining knowledge-based reasoning and machine learning. In *Proceedings of the AAAI 2020 Spring Symposium on Combining Machine Learning and Knowledge Engineering in Practice, AAAI-MAKE*, 2020.

[18] N Guarino and Giancarlo Guizzardi. On the reification of relationships. pages 350 – 357, Red Hook, NY, 2016. Curran Associates, Inc.

[19] Chaojun Han, Fumin Shen, Li Liu, Yang Yang, and Heng Tao Shen. Visual spatial attention network for relationship detection. In *Proceedings of the 26th ACM International Conference on Multimedia*, MM '18, page 510–518, New York, NY, USA, 2018. Association for Computing Machinery.

[20] Pascal Hitzler, Aaron Eberhart, Monireh Ebrahimi, Md Kamruzzaman Sarker, and Lu Zhou. Neuro-symbolic approaches in artificial intelligence. *National Science Review*, 9(6), 2022.

[21] Jena D. Hwang, Chandra Bhagavatula, Ronan Le Bras, Jeff Da, Keisuke Sakaguchi, Antoine Bosselut, and Yejin Choi. Comet-atomic 2020: On symbolic and neural commonsense knowledge graphs. In *AAAI*, 2021.

[22] Rodrigo Toro Icarte, Jorge A. Baier, Cristian Ruz, and Alvaro Soto. How a general-purpose commonsense ontology can improve performance of learning-based image retrieval. In *IJCAI*, pages 1283–1289, 2017.

[23] Filip Ilievski, Pedro Szekely, and Bin Zhang. Cskg: The commonsense knowledge graph. In Ruben Verborgh et al., editors, *The Semantic Web. ESWC 2021*, volume 12731 of *Lecture Notes in Computer Science*. Springer, Cham, 2021.

[24] Youmna Ismaeil, Daria Stepanova, Trung-Kien Tran, Piyapat Saranrittichai, Csaba Domokos, and Hendrik Blockeel. *Towards Neural Network Interpretability Using Commonsense Knowledge Graphs*, pages 74–90. 2022.

[25] Justin Johnson, Agrim Gupta, and Li Fei-Fei. Image generation from scene graphs. In *CVPR 2018*, pages 1219–1228, 2018.

[26] Muhammad Jaleed Khan, John G Breslin, and Edward Curry. Neusyre: Neuro-symbolic visual understanding and reasoning framework based on scene graph enrichment. *Semantic Web*, 2023.

[27] Rabinandan Kishor. Neuro-symbolic ai: Bringing a new era of machine learning. *International Journal of Research Publication and Reviews*, 2022.

[28] Chung-Wei Lee, Wei Fang, Chih-Kuan Yeh, and Yu-Chiang Frank Wang. Multi-label zero-shot learning with structured knowledge graphs. In *CVPR*, pages 1576–1585, 2018.

[29] Vladimir Lifschitz. Answer set planning. In *Proceedings of the 1999 International Conference on Logic Programming*, page 23–37, USA, 1999. Massachusetts Institute of Technology.

[30] Tsung-Yi Lin, Michael Maire, Serge J. Belongie, Lubomir D. Bourdev, Ross B. Girshick, James Hays, Pietro Perona, Deva Ramanan, Piotr Doll'a r, and C. Lawrence Zitnick. Microsoft COCO: common objects in context. *CoRR*, abs/1405.0312, 2014.

[31] Robin Manhaeve, Sebastijan Dumancic, Angelika Kimmig, Thomas Demeester, and Luc De Raedt. Deepproblog: Neural probabilistic logic programming. NIPS'18, page 3753–3763, Red Hook, NY, USA, 2018. Curran Associates Inc.

[32] George A. Miller. WordNet: A lexical database for English. In *Human Language Technology: Proceedings of a Workshop held at Plainsboro, New Jersey, March 8-11, 1994*, 1994.

[33] Neo4j. Neo4j - the world's leading graph database, 2012.

[34] Itthisak Phueaksri, Marc A. Kastner, Yasutomo Kawanishi, Takahiro Komamizu, and Ichiro Ide. Towards captioning an image collection from a combined scene graph representation approach. In *MultiMedia Modeling*, pages 178–190, Cham, 2023. Springer International Publishing.

[35] Robyn Speer and Joshua Chin. An Ensemble Method to Produce High-Quality Word Embeddings (2016). *arXiv e-prints*, page arXiv:1604.01692, 2016.

[36] Robyn Speer, Joshua Chin, and Catherine Havasi. Conceptnet 5.5: An open multilingual graph of general knowledge. *Proceedings of 31St AAAI Conference on Artificial Intelligence*, 2016.

[37] Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. Yago: A core of semantic knowledge. In *Proceedings of the 16th International Conference on World Wide Web*, WWW '07, page 697–706, New York, NY, USA, 2007. Association for Computing Machinery.

[38] Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. CommonsenseQA: A question answering challenge targeting commonsense knowledge. In Jill Burstein, Christy Doran, and Thamar Solorio, editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4149–4158, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

[39] D. Teney, L. Liu, and A. Van Den Hengel. Graph-structured representations for visual question answering. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3233–3241, Los Alamitos, CA, USA, jul 2017. IEEE Computer Society.

[40] Alexandros Vassiliades, Nick Bassiliades, Filippos Gouidis, and Theodore Patkos. A knowledge retrieval framework for household objects and actions with external knowledge. In *SEMANTiCS*, volume 12378 of *LNCS*, pages 36–52, 2020.

[41] Ziwei Wang, Zi Huang, Yadan Luo, and Huimin Lu. Ord: Object relationship discovery for visual dialogue generation. *ArXiv*, abs/2006.08322, 2020.

[42] Zhun Yang, Adam Ishay, and Joohyung Lee. Neurasp: Embracing neural networks into answer set programming. In *International Joint Conference on Artificial Intelligence*, 2020.

[43] Chun-Han Yao, Amit Raj, Wei-Chih Hung, Yuanzhen Li, Michael Rubinstein, Ming-Hsuan Yang, and Varun Jampani. Artic3d: Learning robust articulated 3d shapes from noisy web image collections. 2023.

[44] Ting Yao, Yingwei Pan, Yehao Li, and Tao Mei. Exploring visual relationship for image captioning. In Vittorio Ferrari, Martial Hebert, Cristian Sminchisescu, and Yair Weiss, editors, *Computer Vision – ECCV 2018*, volume 11218 of *Lecture Notes in Computer Science*. Springer, Cham, 2018.

[45] Chuanming Yu, Xiaoli Zhao, Lu An, and Xia Lin. Similarity-based link prediction in social networks: A path and node combined approach. *Journal of Information Science*, 43(5):683–695, 2017.

[46] Yilun Zhou, Steven Schockaert, and Julie Shah. Predicting conceptnet path quality using crowdsourced assessments of naturalness. In *WWW*, pages 2460–2471, 2019.

[47] Jun Zhu, Xianjie Chen, and Alan Loddon Yuille. Deepm: A deep part-based model for object detection and semantic part localization. *ArXiv*, abs/1511.07131, 2015.

[48] Yixin Zhu, Tao Gao, Lifeng Fan, Siyuan Huang, Mark Edmonds, Hangxin Liu, Feng Gao, Chi Zhang, Siyuan Qi, Ying Nian Wu, Joshua B. Tenenbaum, and Song-Chun Zhu. Dark, beyond deep: A paradigm shift to cognitive AI with humanlike common sense. *Engineering*, 6(3):310–345, 2020.

Appendix A

ASP Auxiliary Rules

Following all the auxiliary rules that were formulated through the implementation of the proposed methodology are presented.

A.1 Bounding Box Overlap Ruleset:

1. Rule formulated to calculate the area of a bounding box.

$$\begin{aligned} area(B, A) :-& box(\_, B, Xmin, Ymin, Xmax, Ymax), \\ & A = (Xmax - Xmin) * (Ymax - Ymin). \end{aligned}$$

2. The following rules $overlap/2$, capture the cases concerning whether two bounding boxes overlap with each other.

$$\begin{aligned} overlap(B1, B2) :-& box(\_, B1, Xmin1, Ymin1, Xmax1, Ymax1), \\ & box(\_, B2, Xmin2, Ymin2, \_, \_), \\ & B1 != B2, \\ & Xmin2 \geq Xmin1, \\ & Xmin2 \leq Xmax1, \\ & Ymin2 \geq Ymin1, \\ & Ymin2 \leq Ymax1. \end{aligned}$$

$$\begin{aligned}
overlap(B1, B2) :-\ & box(\_, B1, Xmin1, Ymin1, Xmax1, Ymax1),\\
& box(\_, B2, Xmin2, \_, \_, Ymax2),\\
& B1 != B2,\\
& Xmin2 \geq Xmin1,\\
& Xmin2 \leq Xmax1,\\
& Ymax2 \geq Ymin1,\\
& Ymax2 \leq Ymax1.
\end{aligned}$$

$$\begin{aligned}
overlap(B1, B2) :-\ & box(\_, B1, Xmin1, Ymin1, Xmax1, Ymax1),\\
& box(\_, B2, Xmin2, Ymin2, Xmax2, Ymax2),\\
& B1 != B2,\\
& Xmin1 < Xmax2,\\
& Xmin2 < Xmax1,\\
& Ymin1 < Ymax2,\\
& Ymin2 < Ymax1.
\end{aligned}$$

3. A bounding box $b1$ is candidate part a bounding box $b2$, if its area is covered by the area of $b2$ with a ratio over 75%.

$$\begin{aligned}
candidatePartOf(Bmin, Bmax) :-\ & box(\_, B1, Xmin1, Ymin1, Xmax1, Ymax1, \_),\\
& box(\_, B2, Xmin2, Ymin2, Xmax2, Ymax2, \_),\\
& overlap(B1, B2),\\
& area(B1, A1), area(B2, A2),\\
& Amin = \#min\{A1; A2\},\\
& Amax = \#max\{A1; A2\},\\
& area(Bmin, Amin),\\
& area(Bmax, Amax),\\
& Ymax = \#min\{Ymax1; Ymax2\},\\
& Ymin = \#max\{Ymin1; Ymin2\},\\
& Xmax = \#min\{Xmax1; Xmax2\},\\
& Xmin = \#max\{Xmin1; Xmin2\},\\
& Aovl = (Ymax - Ymin) * (Xmax - Xmin),\\
& 75 <= ((100 * Aovl)/Amin).
\end{aligned}$$

A.2 Knowledge ($meronym$) Enrichment Ruleset

1. The $meronym/3$ relation is the inverse of $hasA/3$.

$meronym(X, Y, C) : -hasA(Y, X, C).$

2. The $meronym/3$ relation is equivalent with the $hasContext/3relation.$

   $meronym(X, Y, C) : -hasContext(X, Y, C).$

3. Transitivity holds between the $meronym$ relations.

$$meronym(X, Y, C) : -meronym(X, Z, C),$$
$$meronym(Z, Y, C).$$

4. The $isSynonymWith/3$ is reflexive.

   $isSynonymWith(X, Y, C) : -isSynonymWith(Y, X, C).$

5. If an object part X is part of a whole object Z, and Z is a subclass of an object Y, then X is part of Y.

$$meronym(X, Y, C) : -meronym(X, Z, C),$$
$$hypernym(Z, Y, C).$$

6. If an object part X is part of a whole object Y, and X is synonym with Z, then Z is also part of Y.

$$meronym(Z, Y, C) : -meronym(X, Y, C),$$
$$isSynonymWith(X, Z, C).$$

7. If an object part X is part of a whole object Y, and Y is synonym with Z, then X is part of Z.

$$meronym(X, Z, C) : -meronym(X, Y, C),$$
$$isSynonymWith(Y, Z, C).$$

A.3 *PartOf* Detection Ruleset

1. A bounding box type is characterized as *Object*, if its label exists in the set of whole object labels $\mathcal{O}$.

   $objectBox(B, L, C) : -label(\_, \_, B, L), object(L, C).$

2. A bounding box type is characterized as *Part*, if its label exists in the set of object part labels $\mathcal{P}$.

   $partBox(B, L, C) : -label(\_, \_, B, L), part(L, C).$

A.4 Error Detection Ruleset

1. Two bounding boxes are spatially related if they belong to different type categories, and a $candidatePartOf/2$ relation between them holds.

$$\begin{aligned} spatial\_partOf(B1, L1, B2, L2) :- & canditatePartOf(B1, B2),\\ & partBox(B1, L1, C),\\ & objectBox(B2, L2, C). \end{aligned}$$

2. Two bounding boxes are semantically related if a $spatial\_partOf/4$ relation between them holds and in the knowledge domain exists a $meronym$ associating their labels.

$$\begin{aligned} semantic\_partOf(B1, L1, B2, L2) :- & spatial\_partOf(B1, L1, B2, L2),\\ & meronym(L1, L2,_) . \end{aligned}$$

3. We consider a that a bounding box was classified by the object detection model with high confidence score, if that score is greater or equal to 0.4. Since ASP does not allow floats, we multiply the confidence score by 1000 (considering up to the third significant digit).

$$\begin{aligned} highConfidence(B) :- & box(\_, B, \_, \_, \_, \_, C),\\ & C \geq 400. \end{aligned}$$

4. The following rule counts how many parts that exceed the confidence threshold, are semantically related with a whole object Y.

$$\begin{aligned} hasNumberOfAccParts(Y, N) :- & N = \#count\{(X, Y) : semantic\_partOf(X, \_, Y, \_),\\ & partBox(X, \_, C, \_), C \geq 400\},\\ & objectBox(Y, \_, \_, \_). \end{aligned}$$

5. We consider a whole object's bounding box as $context$, if it is semantically associated with more than 2 "confident" object parts.

   $context(B) :- hasNumberOfAccParts(B, N),\ N >= 2.$

6. A detected object part that cannot be semantically associated with any whole object in an image, is considered as $single$.

$$\begin{aligned} single(B) :- & partBox(B, \_, \_, \_),\\ & not\ semantic\_partOf(B, \_, \_, \_). \end{aligned}$$

7. For the error detection task, we consider only bounding boxes for whom there is at least one $meronym$ relation in the knowledge domain regarding their classification label. To address that, the following rules were formulated.

   $knownPart(B) : -partBox(B, L, \_, \_), meronym(L, \_, \_).$

   $knownObject(B) : -objectBox(B, L, \_,_{)}, meronym(\_, L, \_).$

A.4.1 Error MO Handling Ruleset

The following rules were produced to address the MO error case. Utilizing them, we are able to find the number of the missing whole objects, as well as their possible classification label.

- Possible class discovery

  1. Infer whether an object part is not a unique part of a whole object, according to our knowledge repository.

     $$\begin{aligned} isNotUniquePart(X) : -&meronym(X, Y),\\ &meronym(X, Z),\\ &Y! = Z. \end{aligned}$$

  2. Infer whether an object part is a unique part of a whole object, according to our knowledge repository.

     $$\begin{aligned} isUniquePartOf(X, Y) : -&part(X), object(Y),\\ &meronym(X, Y),\\ &not\ isNotUniquePart(X). \end{aligned}$$

  3. All the whole object labels that are associated with a $meronym$ relation with an object part that triggered a MO error case, within our knowledge domain, are considered as possible classes.

     $$\begin{aligned} possibleClass(B, Y) : -&single(B),\\ &errorCaseMO(B),\\ &label(\_, \_, B, L),\\ &meronym(L, Y). \end{aligned}$$

  4. If the object part that triggered the MO error is unique $meronym$ of a whole object label, then we report that a whole object with that label is definitely missing.

$$\begin{aligned} detClass(Y) :- & single(B),\\ & label(\_,\_,B,L),\\ & errorCaseMO(B),\\ & isUniquePartOf(L,Y). \end{aligned}$$

- Number of missing whole objects

  1. The following rule counts the number of possible labels that were infered before, regarding a missing whole object.

$$numPosClasses(N) :- N = \#count\{X : possibleClass(\_,X)\}.$$

- We find all the pairs of whole object labels that share at least one object part (according to the knowledge domain).

$$\begin{aligned} sharePart(Y1,Y2) :- & meronym(X,Y1),\\ & meronym(X,Y2),\\ & Y1 != Y2. \end{aligned}$$

- Following, the ASP rules were formulated to group the possible classes into categories.

$$category(1..N) :- numPosClasses(N).$$
$$in\_category(S,C) : possibleClass(\_,S), category(C).$$

The number of categories cannot exceed the number of the possible labels.

$$\begin{aligned} :- & 0\{in\_category(\_,\_)\}N-1,\\ & numPosClasses(N). \end{aligned}$$

Each category must contain labels, that share object parts with the rest of the labels of the current category.

$$\begin{aligned} :- & in\_category(S1,C),\\ & in\_category(S2,C),\\ & S1 != S2,\\ & not\ sharePart(S1,S2). \end{aligned}$$

Two labels cannot belong to different categories, if they share a common part.

$$\begin{aligned}
&:-in\_category(S1, C1),\\
&\quad in\_category(S2, C2),\\
&\quad C1! = C2,\\
&\quad meronym(P, S1),\\
&\quad meronym(P, S2).
\end{aligned}$$

- When the possible classes are grouped into categories, we count the categories and report the total as the (at least) missing object number.

$$numMissingobjects(N) :-N = \#count\{Y : in\_category(\_, Y)\}.$$

Appendix B

Cypher Queries

Below, the Cypher queries that were used through this work are shown. Some queries regard the creation of the subgraphs for each label, while others are used to retrieve the information from the created subgraphs. Note that all the queries were executed through Python.

- The following query is used to create a node in the graph database.

  ```
  CREATE (t:TYPE{label:LABEL})
  ```

- The query presented below is used to find the requested nodes using their labels, and connect them through the given relation.

  ```
  MATCH (o:Object) MATCH(s:Subject)
  WHERE o.label=OLABEL AND s.label=SLABEL
  MERGE (o)-[:RELATION]->(s)
  RETURN *
  ```

- The following query was used to retrieve the nodes connected with a specific edge(relation), from the constructed subgraphs.

  ```
  MATCH (n)-[r:RELATION]->(m) RETURN n,m
  ```

  In our case the queries that were used are:

  ```
  MATCH (n)-[r:HASA]->(m) RETURN n,m
  MATCH (n)-[r:SYNONYM]->(m) RETURN n,m
  MATCH (n)-[r:ISA]->(m) RETURN n,m
  MATCH (n)-[r:HASCONTEXT]->(m) RETURN n,m
  ```

- In order to check if a node exists in our subgraph we deployed the query below. If the label returned is not null, then the node exists.

```
MATCH (u:TYPE{label:LABEL)  RETURN u.label
```

- To count the total of edges that two nodes are connected by, we formulated the presented query.

```
MATCH (n{label:LABEL})-[r]->(m{label:LABEL)
RETURN COUNT(r) AS count
```

- The query showcased below was used to delete specified edges between two nodes of the subgraph.

```
MATCH(n{label:LABEL})-[r:RELATION]->(m{label:LABEL)
DELETE r
```

Appendix C

Part Of ($meronym$) relations in ground truth

All the $meronym$ relations that exist in the Semantic Pascal-Part Ontology, that was provided along with the dataset, are listed in the following tables.

Object	Parts
bicycle	handlebar wheel headlight saddle chain wheel
bus	bodywork door headlight license plate mirror wheel window
potted plant	pot plant

Object	Parts
aeroplane	body engine artifact wing stern wheel
cow	ear horn muzzle tail eye neck torso leg head

Object	Parts
person	arm
	ear
	ebrow
	foot
	hair
	hand
	mouth
	nose
	eye
	neck
	torso
	leg
	head
cat	ear
	tail
	eye
	neck
	torso
	leg
	head
tv monitor	screen
train	coach
	headlight
	locomotive
bird	animal wing
	beak
	tail
	eye
	neck
	torso
	leg
	head
motorbike	handlebar
	headlight
	saddle
	wheel

Object	Parts
car	bodywork
	door
	headlight
	License plate
	mirror
	wheel
	window
horse	ear
	hoof
	muzzle
	tail
	eye
	neck
	torso
	leg
	head
bottle	body
	cap
sheep	ear
	horn
	muzzle
	tail
	eye
	neck
	torso
	leg
	head
dog	ear
	muzzle
	nose
	tail
	eye
	neck
	torso
	leg
	head