Scene Graph Generation by bridging with Knowledge Graphs

EE17B115
Sahadevareddy Adithya Swaroop

Submitted in partial fulfilment
of the requirements for the degree of

Dual Degree (B.Tech and M.Tech)

Department of Electrical Engineering
Indian Institute of Technology Madras
May 2022

Abstract

Scene graphs are powerful representations that parse images into their abstract semantic elements, i.e., objects and their interactions, which facilitates visual comprehension and explainable reasoning. Although several methods have been proposed, the state-of-the-art performance for SGG is still far from acceptable. Most recent advancement achieves only 16% mean recall. On the other hand, commonsense knowledge graphs are rich repositories that encode how the world is structured, and how general concepts interact. So, from using the information which we have in the commonsense graph, we propose a novel graph-based neural network that iteratively propagates information between the two graphs, as well as within each of them, while gradually refining their bridge in each iteration. This message passing through the creation of bridge leads to a better estimate of nodes and predicates, thus improving the mean recall. This Graph Bridging Network, **GB-Net**, successively infers edges and nodes, allowing to simultaneously exploit and refine the rich, heterogeneous structure of the interconnected scene and commonsense graphs.

Use Case: In construction of buildings, it's very difficult to keep track of advancements in work, but by generation of scene graphs from images during construction, we can keep track of everything that's happening. It finally helps in speeding up the process and also data can be well-communicated between employees of different disciplines because the information is digitalized.

Acknowledgements

I wish to express my wholehearted gratitude to my project guide **Dr. V Srinivasa Chakravarthy**, Professor in the Department of Biotechnology, for his valuable advice, technical support from the beginning of my project and giving me a oppourtunity to work on this project. I am grateful for the support provided by **Dr. Mansi Sharma** as the co-guide of this project. Foremost, I would like to express my sincere gratitude to **Krishna Sridhar**, founder and CTO of Conxai Technologies for providing all facilities, timely suggestions and guidance to do this project.

I would like to thank my project mentor **Vigneswaran Chandrasekaran** for all the help and guidance that he has provided. I would also like to thank **Raja Sree** for always assisting me in the project and for being an awesome teammate.

Words are inadequate to express my gratitude to my parents and my brother who have supported me at all times. I always hold a special place in my heart for my grandmother and my uncle who took care of me. My heartiest thanks to the friends who have supported me in all possible ways. I would also like to thank my institution and the faculty members without whom this project would have been a distant reality.

Thesis Certificate

This is to certify that the thesis entitled "**Scene Graph Generation by bridging with Knowledge Graphs**" submitted by **Adithya Swaroop** to the Indian Institute of Technology, Madras for the award of the degree of **Dual Degree (B.Tech and M.Tech)** is a bonafide record of research work carried out by him under my supervision. The contents of this thesis, in full or in parts, have not been submitted to any other Institute or University for the award of any degree or diploma.

Dr. V Srinivasa Chakravarthy
Professor
Department of Biotechnology
Indian Institute of Technology Madras
Chennai – 600 036.

Dr. Mansi Sharma
Professor
Department of Electrical Engineering
Indian Insitute of Technology Madras
Chennai - 600 036

Contents

List of Tables

List of Figures

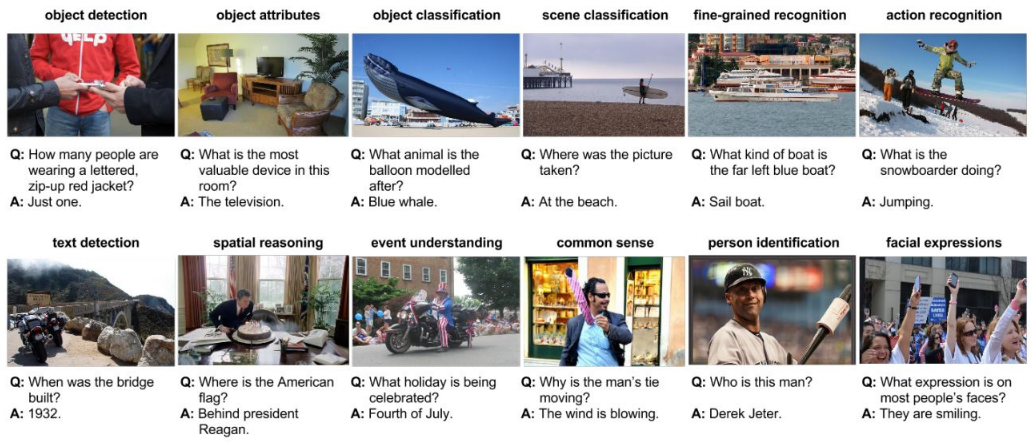
Abbreviations

VG	Visual Genome
GB-Net	Graph Bridging Network
R-CNN	Region based Convolutional Neural Network
SGG	Scene Graph Generation
NLP	Natural Language Processing
VQA	Visual Question Answering
GNN	Graph Neural Network
GRU	Gated Recurrent Unit
MLP	Multi-layered Perceptron
SP	Scene Predicate
SE	Scene Entity
CE	Commonsense Entity
CP	Commonsense Predicate
ROI	Region of Interest
mR	Mean Recall
BIM	Building Information Modelling
AEC	Architecture, Engineering and Construction

Chapter 1

Introduction

Natural language processing (NLP) has a long history of extracting organised, symbolic, semantic representations from data, known as semantic parsing at the sentence level and information extraction at the document level. Many applications exist for the resulting semantic graphs or knowledge graphs, including question answering and information retrieval. **Scene Graph Generation** (SGG), which tries to extract a symbolic, graphical representation from a given image, where each node corresponds to a localised and categorised object (entity), and each edge encodes a paired inter- activity (predicate). Applications of SGG are **Visual Question Answering (VQA), Image captioning, Image synthesis**. Figure 1.1 shows some examples of VQA. Figure 1.2 can let you understand what is SGG, what are commonsense graphs and scene graphs, what are entities and predicates in a basic level.



Q: How many people are wearing a lettered, zip-up red jacket?
A: Just one.

Q: What is the most valuable device in this room?
A: The television.

Q: What animal is the balloon modelled after?
A: Blue whale.

Q: Where was the picture taken?
A: At the beach.

Q: What kind of boat is the far left blue boat?
A: Sail boat.

Q: What is the snowboarder doing?
A: Jumping.

Q: When was the bridge built?
A: 1932.

Q: Where is the American flag?
A: Behind president Reagan.

Q: What holiday is being celebrated?
A: Fourth of July.

Q: Why is the man's tie moving?
A: The wind is blowing.

Q: Who is this man?
A: Derek Jeter.

Q: What expression is on most people's faces?
A: They are smiling.

Figure 1.1: Example QA pairs in the Visual Genome dataset. Source: [3]

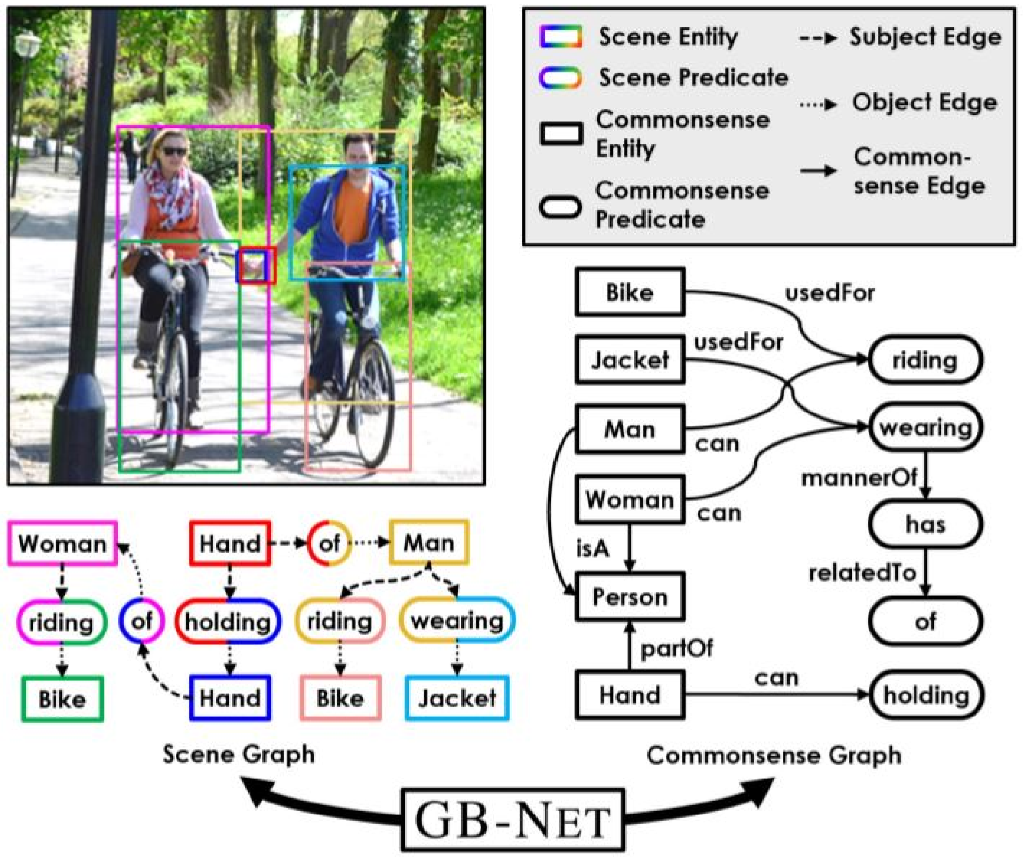


Figure 1.2: **Left:** An example of a Visual Genome image and its ground truth scene graph. **Right:** A section of the commonsense graph that is relevant. Scene Graph Generation is formulated as the problem of constructing a bridge between these two graphs and passing information through it. Such bridge not only classifies each scene entity and predicate, but also creates an inter-connected heterogeneous graph whose rich structure is exploited by this method (GB-Net). Source: [1]

Despite the fact that many methods have been proposed, SGG's current performance is still far from satisfactory. Recently, researchers were only able to match the top 100 predicted subject-predicate-object triples against ground truth triples with a 16 percent mean recall. This shows that the present SGG approaches are insufficient to deal with the task's complexity. The basic idea of those works is that having prior knowledge of the world can be quite useful when interpreting a complex scenario. If we know the relationship of a ***Person*** and a ***Jacket*** is most likely ***wearing***, we can more easily distingush between ***wearing***, ***on***, and ***attachedTo***, and classify their relationship more accurately. Similarly, even if we only observe ***Man-wearing-Jacket*** in training data, we can generalise and recognise a ***Woman-wearing-Jacket*** triplet at test time provided we know a ***Man*** and a ***Woman*** are both sub-types of ***Person***.

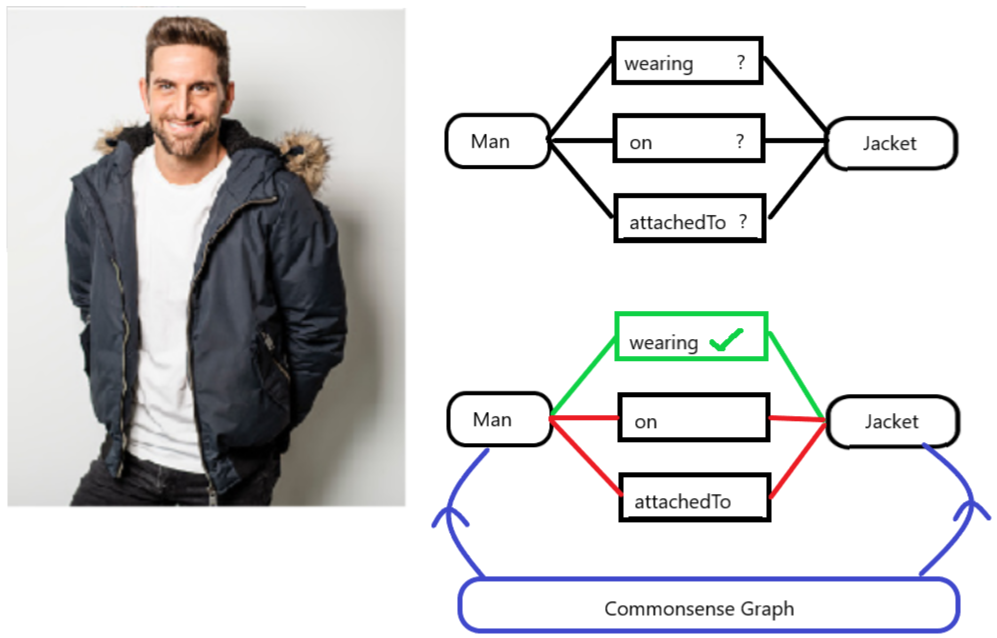


Figure 1.3: Before vs After getting information from Commonsense Graph. All edges are Bidirectional, information can come and enrich the graph, it also can work as a feedback.

Scene and commonsense graphs are both represented as knowledge graphs with entity and predicate nodes and many sorts of edges. A scene graph node represents an entity or predicate instance in a specific image, whereas a commonsense graph node represents an entity or predicate class that is unrelated to the image. A scene graph edge denotes the involvement of an entity instance (such as a subject or object) in a predicate instance in a scene, whereas a commonsense edge denotes a general truth about the interaction of two ideas in the real world.

This challenge of scene graph generation from entity and predicate categorization can be reformed into the innovative problem of bridging the two graphs based on this unified perspective. We are essentially creating a **bridge** between the image and general commonsense, because we realised that from commonsense knowledge we can easily find the links between objects. This creates a link between instance-level visual information and general commonsense knowledge. This method, given an image, initialises possible entity and predicate nodes before classifying each node by connecting it to its appropriate class node in the commonsense graph through a bridge edge. This model is a graph neural network (GNN) that iteratively propagates messages between the scene and commonsense graphs, as well as within each of them, while gradually refining the bridge in each iteration, to incorporate the rich combination of visual and commonsense information in the SGG process. This Graph Bridging Network, GB-Net, infers edges and nodes sequentially, allowing us to leverage and develop the rich, heterogeneous structure of the interconnected scene and commonsense graphs at the same time.

We are also going to learn the use-case of Scene graph generation in **Construction Industry**. It's tough to maintain track of progress in building construction, but by creating scene graphs from photographs throughout construction, we can keep track of everything that's going on. Finally, because the information is digitalized, it aids in the speeding up of the process and allows for effective data communication among personnel from other disciplines. For example, imagine a Virtual Question Answering Protocol (like in Figure 1.1) during the construction of a building or a road. How many bags of cement were transported? Is the crane carrying something? What is the stage of plumbing in fifth floor? Normally, these questions require many people to keep track of the progress and if bigger the project is, more difficult it would be to track. But, GB Net will provide a scalable solution for this issue.

Chapter 2

Graph Neural Networks

Graph Neural Network (GNN) has recently acquired traction in a variety of fields, including social networking, knowledge graphs, recommender systems, and even life science. The ability of GNN to model the dependencies between nodes in a network allows for a breakthrough in graph analysis research. This chapter will explain the fundamentals of Graph Neural Networks as well as an advanced algorithm called GraphSage.

2.1 Graph

Before we get into GNN, let's first understand what is Graph. In Computer Science, a graph is a data structure consisting of two components, vertices and edges. A graph G can be well described by the set of vertices V and edges E it contains.

$$G = (V, E)$$

Depending on whether there are directional relationships between vertices, edges can be either directed or undirected. The vertices are often called nodes.

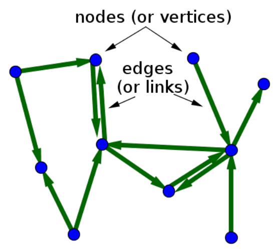


Figure 2.1: Example of a directed graph

2.2 Architecture

A Graph Neural Network is a sort of Neural Network that works with the graph structure directly. Node categorization is a common use of GNN. In essence, each node in the network has a label, and we want to predict the labels of the nodes without using ground-truth.

Each node v in the node classification issue is defined by its feature x_v and is associated with a ground-truth label t_v. The goal is to use the labels of the identified nodes in a partially labelled graph G to predict the labels of the unlabeled nodes. It learns to represent each node with a d dimensional vector (state) h_v that holds information about its neighbors. Where $x_{co[v]}$ denotes the features of the edges connecting with v, $h_{ne[v]}$ means the embedding of the adjacent nodes of v, and $x_{ne[v]}$ denotes the features of the neighbouring nodes of v, and $x_{ne[v]}$ denotes the features of the neighbouring nodes of v. After combining all the information, We may use an iterative update process because we're looking for a unique solution for h_v. Such operation is called as **message passing** because we are passing information(message) from a neighbour to neighbour and formulating the final feature vector h_v for each node from the extracted information.

$$h_v = f(aggregation(x_v, x_{co[v]}, h_{ne[v]}, x_{ne[v]}))$$

The output of the GNN is computed by passing the state h_v as well as the feature x_v to an output function g.

$$o_v = g(h_v, x_v)$$

Both f and g can be viewed as fully-connected feed-forward neural networks. The aggregation method of neighbourhood features can be a mean aggregator for example. The mean aggregator takes the average of the latent vectors of a node and all its neighborhood.

$$h_v^k \leftarrow \sigma(W * mean(h_v^{k-1} \cup h_u^{k-1}, \forall u \in N(v)))$$

The cross-entropy loss can be formulated between the outputs and ground truths of all vertices:

$$loss = \sum_{i=1}^{m}(CE(gt_i, o_i))$$

2.3 Message Passing

Given a heterogeneous graph with different types of nodes and different types of edges (like scene graphs), we need to circulate information among nodes, To compute *outgoing messages*, each node representation is first fed into a fully connected network.

$$\mathbf{m}_i^{\Delta\rightarrow} = \phi_{\text{send}}^{\Delta}(\mathbf{x}_i^{\Delta})$$

ϕ_{send} is a trainable *send head* with weights shared across nodes of each node type Δ, for each node i. After computing outgoing messages, multiply them by the edge weight and transmit them through all outgoing edges. Then, for each node, combine incoming messages by adding edges of the same type first, and then concatenating across edge types. By applying another fully connected network to the aggregated messages, compute the incoming message for each node:

$$\mathbf{m}_j^{\Delta'\leftarrow} = \phi_{\text{receive}}^{\Delta}\left(\bigcup_{\Delta'} \bigcup^{\mathcal{E}_k \in \mathcal{E}^{\Delta'\rightarrow\Delta}} \sum_{(i,j,a_{ij}^k)\in\mathcal{E}_k} a_{ij}^k \mathbf{m}_i^{\Delta'\rightarrow}\right)$$

- ϕ_{receive} is a trainable *receive head* and $\cup$ denotes concatenation.
- first concatenation is over all 4 node types.
- second concatenation is over all edge types from Δ' to Δ.
- the sum is over all edges of that type, where the edge is from i to j.
- a_{ij}^k is the edge weight.

After receiving the message, we should update the nodes by using the information received. If we use a Gated Recurrent Unit (GRU, a type of RNN) update rule,

$$\mathbf{z}_j^{\Delta} = \sigma(W_z^{\Delta}\mathbf{m}_j^{\Delta\leftarrow} + U_z^{\Delta}\mathbf{x}_j^{\Delta})$$

$$\mathbf{r}_j^{\Delta} = \sigma(W_r^{\Delta}\mathbf{m}_j^{\Delta\leftarrow} + U_r^{\Delta}\mathbf{x}_j^{\Delta})$$

$$\mathbf{h}_j^{\Delta} = tanh(W_h^{\Delta}\mathbf{m}_j^{\Delta\leftarrow} + U_h^{\Delta}(\mathbf{r}_j^{\Delta} \odot \mathbf{x}_j^{\Delta}))$$

$$\mathbf{x}_j^{\Delta} \Leftarrow (1 - \mathbf{z}_j^{\Delta}) \odot \mathbf{x}_j^{\Delta} + \mathbf{z}_j^{\Delta} \odot \mathbf{h}_j^{\Delta}$$

W^{Δ} and U^{Δ} are matrices being trained and they are distinct for each node type Δ. This update rule can be thought of as a heterogeneous graph extension of GNN, with

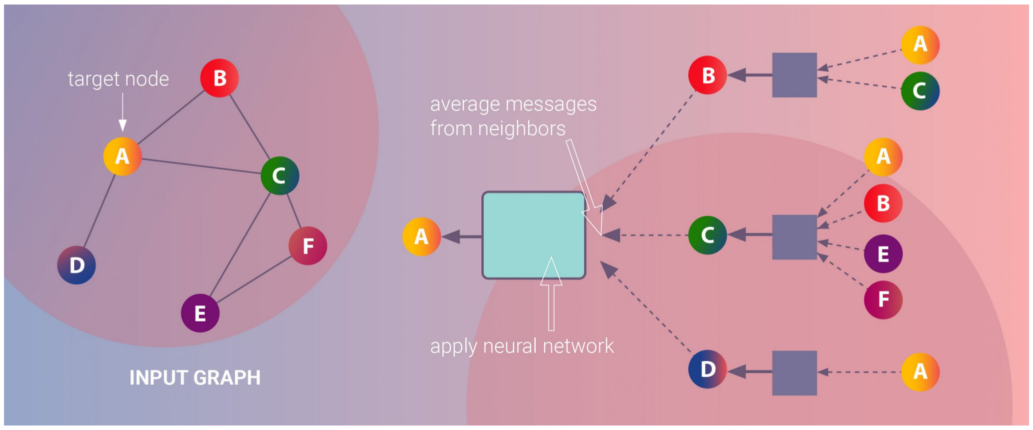


Figure 2.2: Illustration of Message Passing. Source: A blog on GNN

a more complicated message aggregation approach compared to mean aggregation strategy discussed in previous section.

The number of layers in a graph neural network determines the maximum distance that messages from each node can travel over the data structure's connections. We'll run message passing twice in a two-layer network, for example, so the signal will only travel two hops from the source node and won't be affected by information from outside the subgraph. We must use extra layers if long-range dependencies are critical to the situation at hand.

2.4 Comparison with traditional Convnets

If any of this sounds familiar, it's because it is: We generally slide a convolutional operator across a pixel grid in standard convnets and process one sub-patch of an image at a time. We combine and transform pixel value information to create new representations, and, in a way, we treat groups of pixels as neighborhoods of nodes. Overall, the logic of GCNs is very similar to that of all conventional neural nets: one layer's output is passed as input to the next layer. There are, nevertheless, some significant differences.

1. We run the preprocessing step of aggregating and averaging the values of a node's neighbours on every layer in GCNs.

2. A node's computational graph will be defined by its local network neighbourhood, which will be distinct from that of all other vertices. As a result, each

node will have its own neural network design that will capture and reflect its environment's structure.

2.5 Limitations

Despite the tremendous breakthroughs GNNs have made on particular datasets (state-of-the-art performances on node classification, link prediction, and graph classification), there are still flaws that the community is working to resolve. The following are the two main concerns:

1. In general, GNNs are not noise-resistant. We can reverse all of the classifications that a GNN produces with simple feature perturbations and edge addition (or deletion).

2. If their input features are homogeneous, GCNs can sometimes fail to distinguish between basic graph architectures with slightly varied adjacency matrices. In downstream tasks like graph classification, this poses a significant challenge.

2.6 Conclusion

A graph neural network is like a normal MLP, but in each layer, weight matrix is considered only between the nodes where edge is present in the graph. The main idea of MPNN (Message Passing Neural Network) framework consists of message, update, and readout functions which operate on different nodes in the graph. A GNN is employed as an engine for a recommendation system at Pinterest, for example, and it provides by far the best embeddings, according to Jure Leskovec (Pinterest's Chief Scientist). Uber, too, recently released a blog post outlining how a GNN-based strategy helps them provide accurate UberEats suggestions. In future, most of the databases will be graphs and GNNs are going to make a huge impact in deep learning. The power of GNN in modeling complex graph structures is truly astonishing.

Chapter 3

Representation of Knowledge Graphs

For GNN computations, any type of databases should be formulated as graphs. Similar to all other graphs, we need to define edges, nodes for Knowledge Graphs too. Any Knowledge graph can be defined as a set of entity and predicate nodes and set of directed and weighted edges. Each node will have a semantic label. The two types of Knowledge Graphs which we try to represent mathematically are Scene Graphs and Commonsense Graphs.

3.1 Commonsense Graph

Recollect that commonsense knowledge graphs are rich repositories that **encode how the world is structured, and how general concepts interact**. We represent the predicate nodes of Commonsense Graph as Commonsense Predicates (CP) and entity nodes of Commonsense entities (CE). Each node represent the general concept of its semantic label, there won't be any repetition of nodes. Each edge encodes a relational fact involving a pair of entities with a predicate. For example, ***Man-can-Eat, Wheel-partOf-Vehicle*** consists of pair of two concepts and how they are related. So each relation $CE \rightarrow CP \rightarrow CE$ in a commonsense graph represented by two edges: $CE \rightarrow CP$ and $CP \rightarrow CE$, but also there exists possibility of edges like $CP \rightarrow CP$ or $CE \rightarrow CE$.

For a set of edges encoding the relation r between a commonsense graph nodes,

$$\mathcal{E}_C = \{\mathcal{E}_r^{CE \rightarrow CP}\} \cup \{\mathcal{E}_r^{CP \rightarrow CE}\} \cup \{\mathcal{E}_r^{CE \rightarrow CE}\} \cup \{\mathcal{E}_r^{CP \rightarrow CP}\}$$

3.2 Scene Graph

Similar to Commonsense graph, Scene graph contains Scene Entity(SE) nodes and Scene Predicate(SP) nodes, but it's slightly different. Recollect that **Scene Graphs parse images into their abstract semantic elements, i.e., objects and their interactions, which facilitates visual comprehension**. Scene graph contains the following information:

- Each SE node is associated with a bounding box on the image.
- Each SP node is associated with an ordered pair of SE nodes, namely a subject and an object.
- Since there are two types of entity nodes: subjects and objects, there are four types of edges: **sub-pred, pred-sub, obj-pred, pred-obj.**

$$\mathcal{N}_{SE} \subseteq [0,1]^4 \text{ X } \mathcal{N}_{CE}$$

$$\mathcal{N}_{SP} \subseteq \mathcal{N}_{SE} \text{ X } \mathcal{N}_{CP} \text{ X } \mathcal{N}_{SE}$$

$$\mathcal{E}_S = \{\mathcal{E}^{SE \to SP}_{subjectOf}\} \cup \{\mathcal{E}^{SE \to SP}_{objectOf}\} \cup \{\mathcal{E}^{SP \to SE}_{hasSubject}\} \cup \{\mathcal{E}^{SP \to SE}_{hasObjject}\}$$

$[0,1]^4$ is a set of possible 4-pairs of coordinates each constituting a bounding box. $\mathcal{N}_{SE}$ X $\mathcal{N}_{CP}$ X $\mathcal{N}_{SE}$ denotes all possible triplets of two SE nodes and a SP node.

3.3 Bridging Problem Statement

What we have to do:

- Predict the label for every bounding box with a subject or object (We use a Faster R-CNN for object detection)
- Predict the predicate for every possible pair of scene entity nodes. (ignore if there is no possibility for a particular pair)

How can we do:

- By introducing a set of **bridge** edges $\mathcal{E}_B$ that connect each SE node to it's corresponding CE node (creating a *bridge* between Scene and Commonsense graphs for information sharing).

$$\mathcal{N}^{?}_{SE} \subseteq [0,1]^4$$

$$\mathcal{N}^{?}_{SP} \subseteq \mathcal{N}_{SE} \text{ X } \mathcal{N}_{SE}$$

$$\mathcal{E}_B = \{\mathcal{E}^{SE \to CE}_{classifiedTo}\} \cup \{\mathcal{E}^{SP \to CP}_{classifiedTo}\} \cup \{\mathcal{E}^{CE \to SE}_{instanceOf}\} \cup \{\mathcal{E}^{CP \to SP}_{instanceOf}\}$$

The superscript $^{?}$ mean that the predicted label is removed from the SE node and the label is unknown. Though the label is removed, it's connected to the corresponding node in the commonsense graph. There are two types of bridge edges: ***classifiedTo*** and ***instanceOf***. Edges in the direction of scene graph to commonsense graph are called ***classifiedTo*** and ***instanceOf*** is other way around. This is because, CE nodes are considered as labels and SE nodes are considered as instances of those labels.

Given an image I and a commonsense graph with required knowledge, the goal of SGG is to maximize:

$$\mathbf{p}(\mathcal{N}_{SE}, \mathcal{N}_{SP}, \mathcal{E}_S | I, \mathcal{N}_{CE}, \mathcal{N}_{CP}, \mathcal{E}_C) =$$

$$\mathbf{p}(\mathcal{N}^{?}_{SE}, \mathcal{N}^{?}_{SP}, \mathcal{E}_S | I) \text{ x } \mathbf{p}(\mathcal{E}_B | I, \mathcal{N}_{CE}, \mathcal{N}_{CP}, \mathcal{E}_C, \mathcal{N}^{?}_{SE}, \mathcal{N}^{?}_{SP}, \mathcal{E}_S)$$

First expression of RHS is maximized by implementing a region proposal network that generate bounding boxes given an image. Second term is maximized by implementing this Graph Bridging network (GB-Net) that creates bridge edges between commonsense graph and scene graph.

Chapter 4

Method

4.1 Introduction

Given an image, the proposed method is:

1. Apply Faster R-CNN and detect objects.
2. Represent them as scene entity (SE) nodes.
3. Apply a simple predicate proposal algorithm, which is creating a scene predicate (SP) node for each pair of SE nodes, yet to be classified.
4. Combine this with a background commonsense graph. We generate bridge edges between the two networks that connect each instance (SE and SP node) to its corresponding class (CE and CP node).
5. Messages should be propagated among all nodes, via each edge type, with dedicated message passing parameters.
6. Compute pairwise similarity between every SP node and every CP node, and discovers most similar pairs to match scene predicates to their respective classes, using the updated node representations after message passing.
7. Similarly we can compare similar SE and CE nodes for better predictions for entity nodes.
8. Repeat the process of propagating messages on refined bridges till final steady state is achieved, in which we can extract output scene graph.

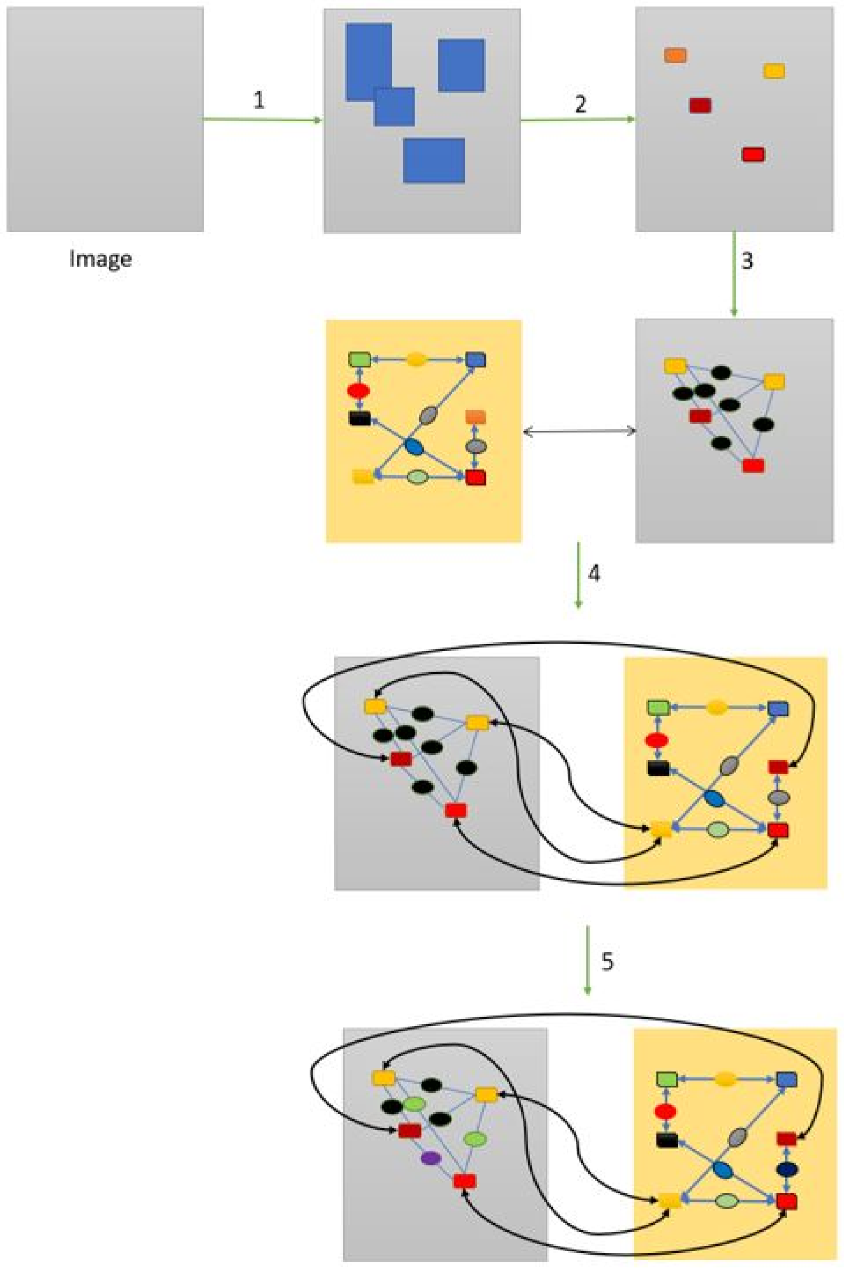


Figure 4.1: Pipeline of scene graph generation. Grey background graph represent the generating scene graph and yellow background graph represent commonsense graph. In step-4, similar nodes are connected by *bridge* edges. In step-5, similar predicate nodes between scene graph and commonsense graph are found and color coded.

4.2 Features

4.2.1 Scene Graph

The object detection network (Faster R-CNN) outputs a set of n detected objects. Each of those will have a output distribution p_i, a bounding box b_i, and an ROI-

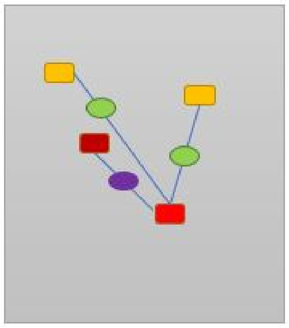


Figure 4.2: Final scene graph after removal of unfamiliar predicates by comparing predicate node features of both knowledge graphs

aligned feature map v_i. Every scene entity node feature vector is derived from this feature vector.

$$x_i^{\text{SE}} = \phi_{\text{init}}^{\text{SE}}(v_i)$$

where $\phi_{\text{init}}^{\text{SE}}$ is a fully connected layer branched from the backbone (Faster R-CNN) after ROI-align.

Recollect our simple predicate proposal algorithm: creating scene predicates between every pair of SE nodes. Feature vectors of SP nodes, say u_i, are initialized by using the ROI features of the bounding box enclosing the union of it's subject and object. So, suppose $\phi_{\text{init}}^{\text{SP}}$ is also a fully connected layer branching from the backbone, then

$$x_i^{\text{SP}} = \phi_{\text{init}}^{\text{SP}}(u_i)$$

Coming to the edges, as we discussed before, there are four types of edges: $s \to p$ (**subjectOf**), $p \to s$ (**hasSubject**), $o \to p$ (**objectOf**), $p \to o$ (**hasObject**). Connect each predicate node to it's corresponding subject and object such that each triplet $s \to o \to p$ has 4 edges among them. The reason for having two different types of edges for opposite directions is that how we use predicate information to update entities in the message passing phase should be different from how we use entities to update predicates.

4.2.2 Commonsense graph

For commonsense graph nodes, we can directly extract features from their word embeddings, as the embeddings already will have semantic knowledge of that word. So we can initialize the feature vector for commonsense graph nodes as linear projection of their embeddings.

$$x_i^{\text{CE}} = \text{proj}_{\text{init}}^{\text{CE}}(e_i^{\text{ent}}) \quad \text{and} \quad x_i^{\text{CP}} = \text{proj}_{\text{init}}^{\text{CP}}(e_i^{\text{pred}})$$

Commonsense graph has various types of edges like **usedFor, partOf, relatedTo, isA, mannerOf, similarTo**.

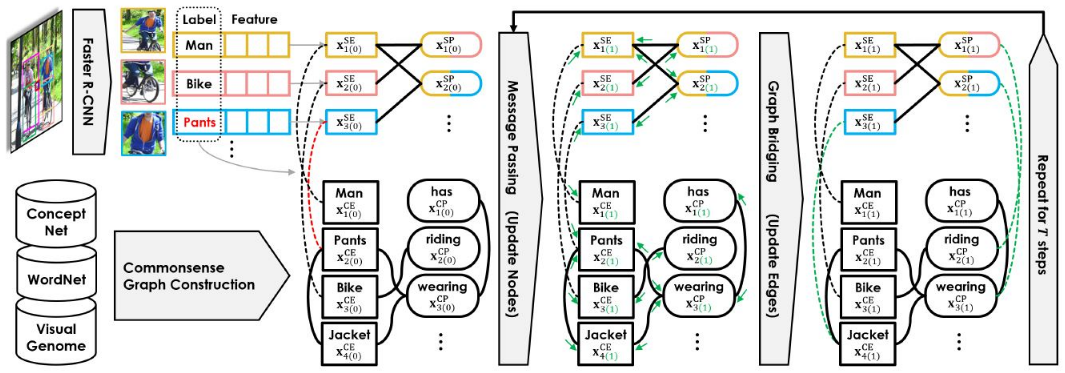


Figure 4.3: An illustration of the GB-Net procedure. Using a Faster R-CNN, we first initialise the scene graph and entity bridges. Messages are then propagated to update node representations, which are then used to update entity and predicate bridges. This is done T times, with the last bridge determining each node's output label. Source: [1]

4.2.3 Bridge edges

There are two types of bridge edges, ***classifiedTo*** and ***instanceOf***. An SE node representing a detected **Cat** intuitively refers to the ***Cat*** concept, and thus the commonsense graph's ***Cat*** node. So, we use a ***classifiedTo*** edge type to connect each SE node to the CE node that corresponds to the label predicted by Faster R-CNN. Instead of one-to-one classification, intuitively it makes sense to connect each entity to top K_{bridge} classes using label distribution p_i as weights.

A new hasInstance edge to construct a reverse link from each CE node to the associated SE nodes, but with the same weights. This, as previously said, is to ensure that information flows from commonsense to scene as well as scene to commonsense, but not in the same way.

For predicates, define two new edge types for predicates, classifiedTo and hasInstance, which are initially empty and will be updated to connect SP and CP nodes.

These four edge types can be thought of as flexible bridges that connect the two fixed graphs, which are latent variables that the model must determine.

This results in a graph with four different types of nodes (SE, SP, CE, and CP) and different types of edges: scene graph edges E_S like **hasObject**, commonsense edges $\mathcal{E}_C$ like **isA**, and bridge edges E_B like **classifiedTo**. Next, we'll discover how to keep commonsense and scene edges fixed while updating node representations and bridge edges.

4.3 Message passing and bridging

We apply the concepts of message passing from Chapter 2, Section 3 as this is a heterogeneous graph with different types of nodes and connected by different types of edges. Using the equations, we can find the updated node representations for each iteration. We should now update the bridge edges $\mathcal{E}_B$ that connect scene nodes to commonsense nodes using the revised node representations. To accomplish this, we compute a pairwise similarity between each SE and all CE nodes, as well as between each SP and all CP nodes.

$$\mathbf{a}_{ij}^{EB} = \frac{exp\langle \mathbf{x}_i^{SE}, \mathbf{x}_j^{CE} \rangle_{EB}}{\Sigma_{j'} exp\langle \mathbf{x}_i^{SE}, \mathbf{x}_{j'}^{CE} \rangle_{EB}}, \text{ where } \langle \mathbf{x}, \mathbf{y} \rangle_{EB} = \phi_{att}^{SE}(\mathbf{x})^T \phi_{att}^{CE}(\mathbf{y})$$

$$\mathbf{a}_{ij}^{PB} = \frac{exp\langle \mathbf{x}_i^{SP}, \mathbf{x}_j^{CP} \rangle_{PB}}{\Sigma_{j'} exp\langle \mathbf{x}_i^{SP}, \mathbf{x}_{j'}^{CP} \rangle_{EP}}, \text{ where } \langle \mathbf{x}, \mathbf{y} \rangle_{PB} = \phi_{att}^{SP}(\mathbf{x})^T \phi_{att}^{CP}(\mathbf{y})$$

where ϕ_{att}^{Δ} is a fully connected network that resembles *attention head* and it's different for each node type. We use each $\mathbf{a}_{ij}$ obtained as edge weights for bridge edges like ***hasInstance*** and ***classifiedTo***, for both entity bridges and predicate bridges. Given the updated bridges, propagate messages again to update node representations, and iterate this process for a fixed number of steps. The final values of $\mathbf{a}_{ij}^{EB}$ and $\mathbf{a}_{ij}^{PB}$ are the outputs of our model, which can be used to classify each entity and predicate in the scene graph.

4.4 Class-balanced loss

Because of the highly skewed predicate statistics in dataset, the best-performing models tend to predict the predicate nodes as the most common classes, such as on and wearing. To overcome this, we use a recently proposed class-balanced loss, in which each class's weight is inversely proportional to its frequency. For each of the predicate nodes, we use the following loss function:

$$\mathcal{L}_i^P = -\frac{1-\beta}{1-\beta^{n_j}}\ log\ \mathbf{a}_{ij}^{PB}$$

This can be seen as an upgrade to traditional cross-entropy loss. Here j is the class index of the ground truth predicate aligned with i, n_j is the frequency of class j in training data, and β is a hyper-parameter. By observing, we can realise that setting β to 0 will make it a regular cross-entropy loss and if β tends to 1, it suppresses more frequent classes and making sure the network learn the predicates well.

Chapter 5

Implementation and Results

5.1 Dataset

5.1.1 Images and Labels

We use Visual Genome Dataset to train this network. Visual Genome offers a multi-layered perspective on images. It enables a multi-perspective examination of an image, ranging from pixel-level information such as objects to relationships that necessitate more inference, and even deeper cognitive tasks like as question answering. It's a large dataset that can be used to train and assess the next generation of computer vision models. This contains 108,077 photos with labelled objects (entities) and paired relationships (predicates), which are then post-processed to generate scene graphs. The most frequent 150 entity classes and 50 predicate classes are filtered out to create a new sub-dataset for training this network.

5.1.2 Commonsense Graph

This approach makes use of a background knowledge graph to store commonsense information about the target entity and predicate classes. Such data has been shown to be useful in the creation of scene graphs. This makes intuitive sense because they can aid the model in distinguishing between different visual classes by utilizing higher-level semantic meanings and class relationships. Ontological hierarchy is another type of useful information for scene graph generation from Commonsense graph. If we know the relationship of a ***Person*** and a ***Jacket*** is most likely ***wearing***, we can more easily distingush between ***wearing***, ***on***, and ***attachedTo***, and classify their relationship more accurately.

Man and **Woman** are both subtypes of **Person**, therefore a typical person can generalise **Man**'s traits to **Woman**'s. Another example is statistical commonsense, in which the average human understands that a **Vase** is more likely to be on a **Table** than anywhere else, not because of semantic features, but because that is how things are usually done.

Commonsense graph has 150 entities and 50 predicates as conventionally used in the SGG task. We compile the edges (i.e. facts) from a variety of datasets and APIs like WordNet, ConceptNet, to make it as rich and comprehensive as possible. Overall, these three sources lead to 19 edge types (including backward edge types for asymmetric relationships).

5.2 Implementation

The main goal is to jointly infer entities and predicates from scratch given an image. We need to create a task to more clearly evaluate entity and predicate recognition because this task is restricted by the quality of the object proposals. So, we take object detection for granted and provide the model with both ground truth bounding boxes and their true entity class. This is partly because object detection performance is a bottleneck that limits the performance. The average per-image recall of the top K subject-predicate-object triplets is the major evaluation parameter in each challenge. The classification confidence of all three factors is multiplied to calculate the confidence of a triplet that is used for ranking. Each predicate in the ground truth scene graph creates a triplet, which we compare to the top K triplets in the output scene graph. If all three items are correctly identified, a triplet is matched.

For all $\phi_{init}, \phi_{send}, \phi_{recieve}, \phi_{att}$ networks discussed before, we use three-layer fully connected networks with ReLU activation. We perform only 3 message passing steps and node representations are 1024-dimension. For object detection, we use Faster R-CNN which has a VGG-16 backbone and predicts 128 proposals. After many trails of hyperparameter tuning, the best values of β(Section 4.4) is 0.999 and best value of K_{bridge}(Section 4.2.3) is 5. We use GloVE embeddings for entity node initialization in Commonsense graph.

5.3 Results

Before comparing results, let us know about some existing methods in SGG. Most basic method is **FREQ**, it is a simple baseline model that uses statistics solely from the training data to predict the most frequent predicate for any given pair of entity classes. **IMP+** employs a Faster R-CNN backbone, however it doesn't use any external information, instead relying on message passing between entities and predicates to classify them. Still FREQ outperforms IMP+, which implies commonsense knowledge is more important. There's also **SMN**, which applies bi-directional LSTMs on top of entity characteristics before classifying each entity and pair, and **KERN**, which encodes VG statistics into the graph's edge weights before propagating messages.

Mean Recall(mR): We compute the recall for each predicate class separately and take the mean. It's worth mentioning that mean recall is more essential than overall recall, because most SGG approaches tend to achieve high overall recall by focusing on the few most common classes and disregarding the rest. In **mR** metric, all predicates are given equal method in deciding if the method is good or not.

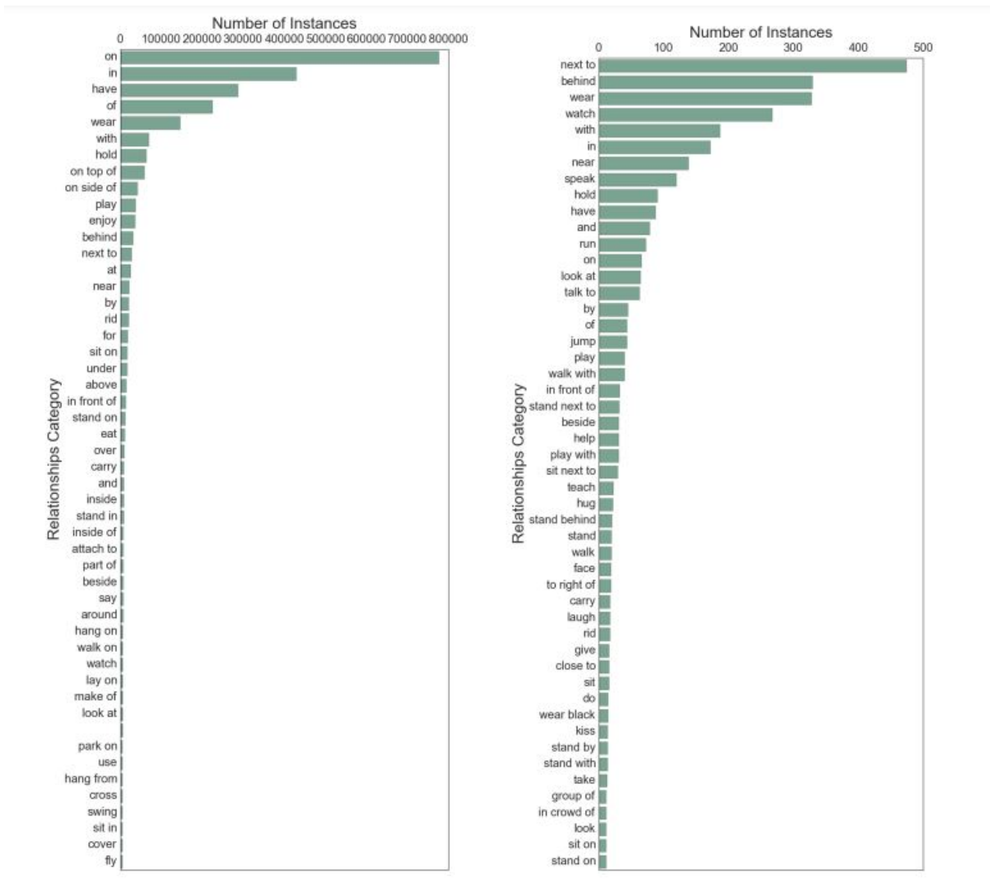


Figure 5.1: Some of most frequent relations in Visual Genome. Mean recall is more essential than overall recall, because most SGG approaches tend to achieve high overall recall by focusing on the few most common classes and disregarding the rest.

We can observe that GB-Net-β model improves mean-recall significantly. If mean-

Table 5.1: Evaluation in terms of mean and overall recall, for top 50 and top 100 triplets according to classification confidence, with and without class-balance loss parameter (β). Numbers are in percentage. The best method for each metric is shown in **bold**.

Metric	Method					
	FREQ	IMP+	SMN	KERN	GB-Net	GB-Net-β
R@50	71.3	75.2	81.1	81.9	82.87	**83.45**
R@100	81.2	83.6	88.3	88.9	89.95	**90.31**
mR@50	24.8	20.3	27.5	36.3	39.59	**44.52**
mR@100	37.3	28.9	37.9	49.0	53.38	**58.65**

recall is high for a method, we can assume that it is less-biased, Also, this model has better recall compared to any other model performing scene graph generation.

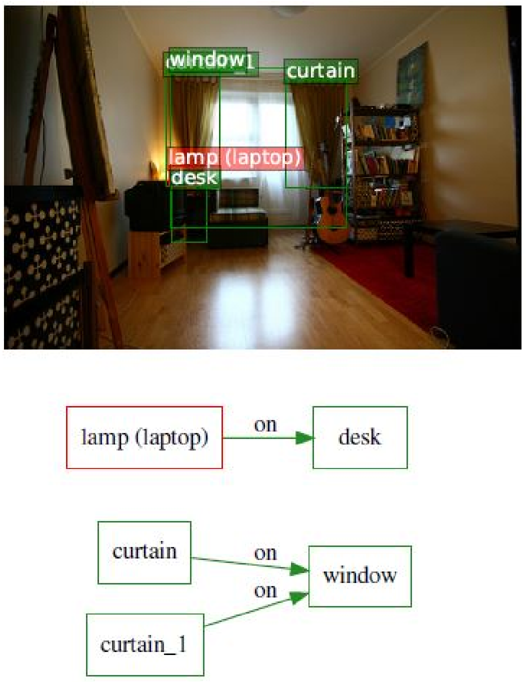


Figure 5.2: Example image and it's scene graph. Source :[1]

Chapter 6

Application in Construction

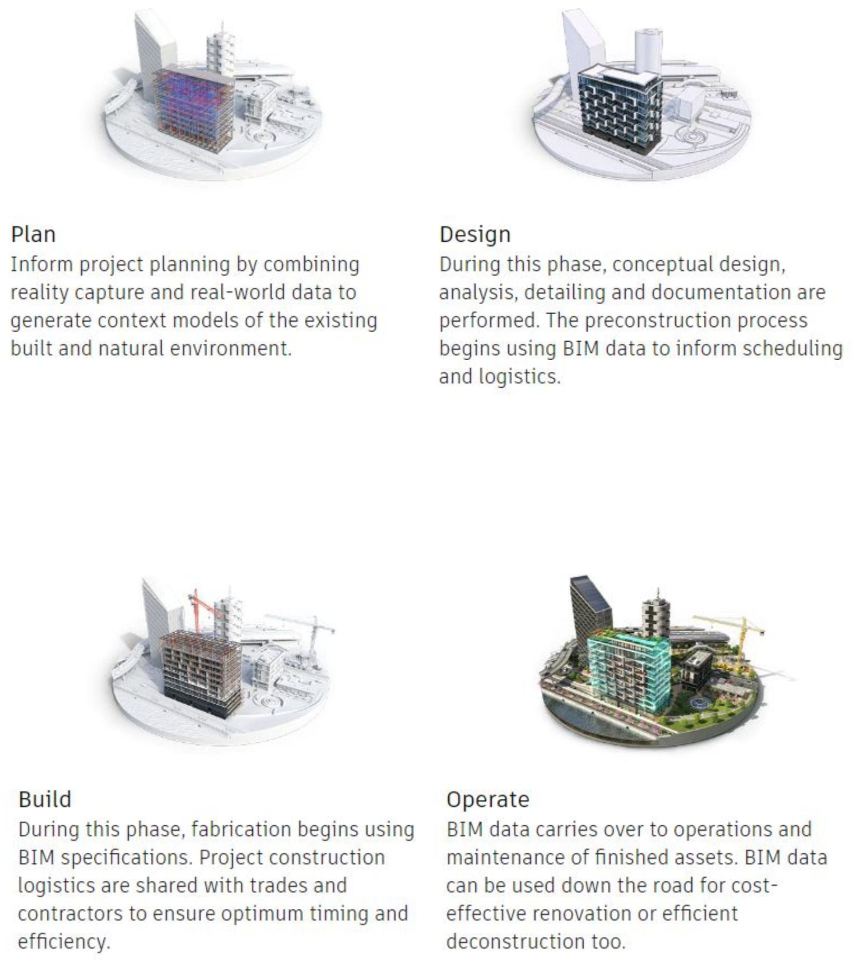
According to the United Nations, the world's population will reach 9.7 billion people by 2050. Not only must the global AEC (Architecture, Engineering, and Construction) industry turn to smarter, more efficient ways to design and build to keep up with global demand, but also to help develop smarter, more resilient spaces.

Despite the fact that AI is becoming more widely used in all fields, it has had little impact on the AEC business. (At least not in comparison to other businesses such as medicine, agriculture, and so on.) This is due to a lack of well-organized data, as there are numerous disciplines such as mechanical (heating, ventilation, and air conditioning, or HVAC for short), electrical, and plumbing. It's challenging to bring together data from a variety of fields in one place. We can, however, parse the data to the ML model we want to train as we need with the help of BIM (Building Information Modelling) and adequate preprocessing.

6.1 BIM introduction

The complete process of creating and managing information for a constructed asset is known as Building Information Modeling (BIM). BIM integrates structured, multi-disciplinary data to build a digital 3D representation of an asset across its lifecycle, from planning and design to construction and operations, using an intelligent model and a cloud platform.

It's tough to maintain track of progress in building construction, but by creating scene graphs from photographs throughout construction, we can keep track of everything that's going on. Finally, because the information is available digitally, we can use the information achieved by SGG to update the BIM model, any change that is being done can be automatically modelled back into the data without any interference



Plan
Inform project planning by combining reality capture and real-world data to generate context models of the existing built and natural environment.

Design
During this phase, conceptual design, analysis, detailing and documentation are performed. The preconstruction process begins using BIM data to inform scheduling and logistics.

Build
During this phase, fabrication begins using BIM specifications. Project construction logistics are shared with trades and contractors to ensure optimum timing and efficiency.

Operate
BIM data carries over to operations and maintenance of finished assets. BIM data can be used down the road for cost-effective renovation or efficient deconstruction too.

Figure 6.1: BIM Usage (Source: Autodesk)

of any third party, thus reducing the resources, time taken.

6.2 Updating BIM by generating Scene graphs

Now, even though BIM is good in handling data, it has lot of limitations. For example, because it contains data from different disciplines, frequently updating it is a big problem. Any minor mistake may generate clashes with other departments. So, using SGG, we can automate the process of BIM update Suppose, a camera captures this incident (Figure 6.1). From that picture, if the model realises the wheelbarrow is transferred, digital data can be changed (for example, amount of cement used can be tracked). Also, we can learn that building already has a foundation. Okay, but for all

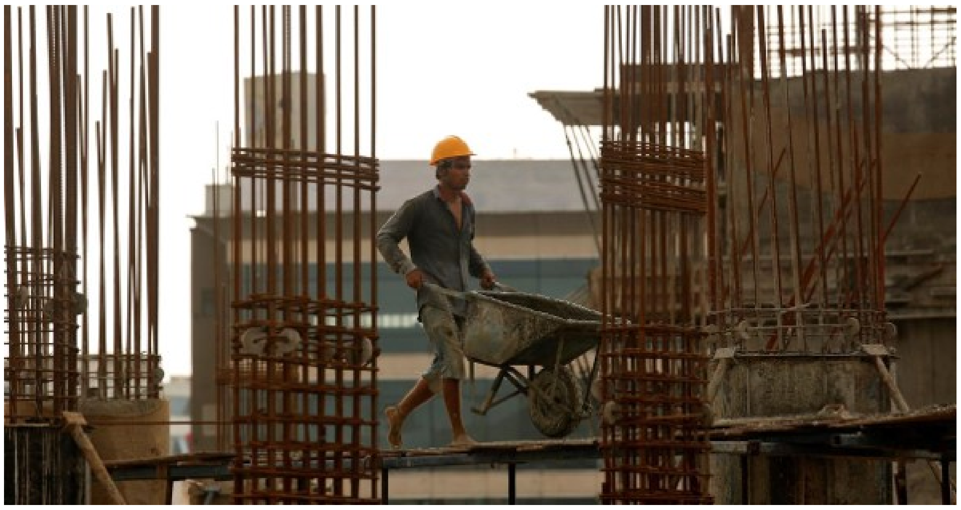


Figure 6.2: A typical photograph taken by a camera during the construction.

of this, we need a new commonsense graph with some construction terms and with information about how buildings, construction process work in real life. Figure 6.2 gives an example illustration of Commonsense graph in Construction Domain. Now if we retrain the network using the images in construction domain, (or we can use transfer learning approach) we can achieve that target. Some examples of images and their output scene graphs too are illustrated below (Fig 6.4, 6.5, 6.6, 6.7).

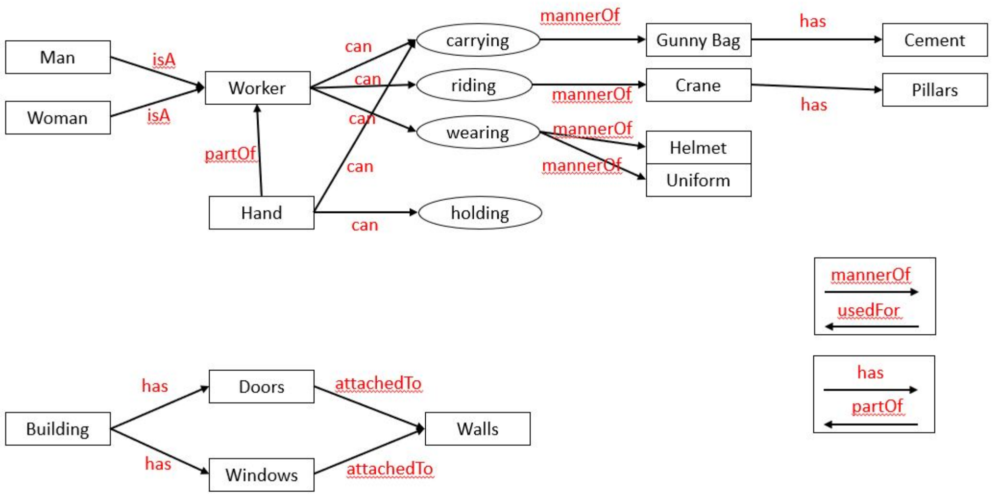


Figure 6.3: Example Commonsense graph in construction domain.

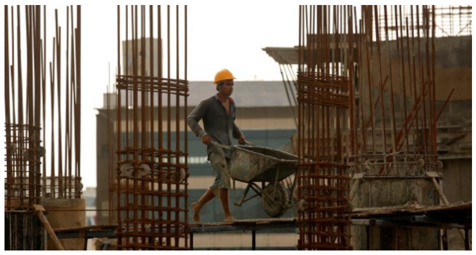


Figure 6.4: A worker carrying a wheelbarrow

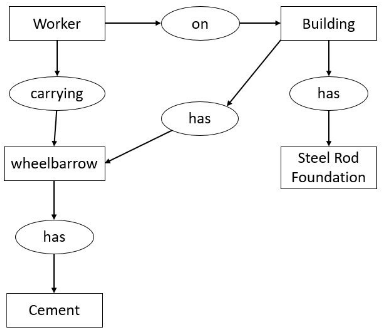


Figure 6.5: Scene graph

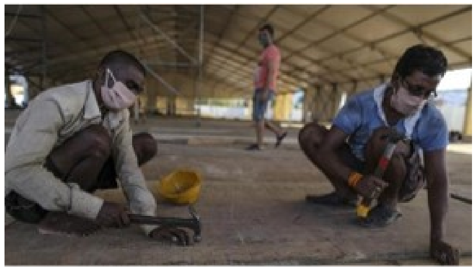


Figure 6.6: Two workers holding hammers

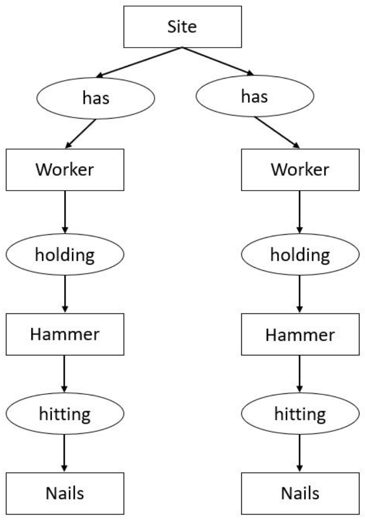


Figure 6.7: Scene graph

Chapter 7

Conclusion

This unique, graphical neural architecture includes external commonsense knowledge in a bridge new way for Scene Graph Generation. We combined the scene graph and commonsense graph formulations into two types of knowledge graphs that are fused into a single graph using a dynamic message passing and bridging mechanism. This approach propagates messages to update nodes in an iterative manner, then compares nodes to update bridge edges, and so on until the two graphs are carefully connected. In various metrics, this technique outperforms the current state of the art.

Coming to the use-case in construction domain, if we are successfully able to parse data digitally, and update BIM, it will be a game changer, AEC industry generally suffers with loss of resources due to lack of automation in updating the status and this solves it. Also, there is no room for any corruption in this case and there wont be any catastrophes.

References

1. Alireza Zareian, Svebor Karaman, Shih-Fu Chang: *Bridging Knowledge Graphs to Generate Scene Graphs*, Jan 2020

2. Ting Yao, Yingwei Pan, Yehao Li and Tao Mei: *Exploring Visual Relationship for Image Captioning*, Sep 2018

3. Ranjay Krishna , Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen, Yannis Kalantidis, Li-Jia Li, David A. Shamma, Michael S. Bernstein, Li Fei-Fei : *Visual Genome, Connecting Language and Vision Using Crowdsourced Dense Image Annotations*

4. Yang, X., Tang, K., Zhang, H., Cai, J.: *Auto-encoding scene graphs for image captioning*, 2019

5. Xu, H., Jiang, C., Liang, X., Lin, L., Li, Z.: *Reasoning-rcnn: Unifying adaptive global reasoning into large-scale object detection*, 2019