Improved Scoring Models for Semantic Image Retrieval Using Scene Graphs

by

Erik T. Conser

A thesis submitted in partial fulfillment of the
requirements for the degree of
Master of Science
in
Computer Science

Thesis Committee:
Melanie Mitchell, Chair
Feng Liu
Bart Massey

Portland State University
2017

Abstract

Image retrieval via a structured query is explored in Johnson, et al. [7]. The query is structured as a scene graph and a graphical model is generated from the scene graph's object, attribute, and relationship structure. Inference is performed on the graphical model with candidate images and the energy results are used to rank the best matches. In [7], scene graph objects that are not in the set of recognized objects are not represented in the graphical model. This work proposes and tests two approaches for modeling the unrecognized objects in order to leverage the attribute and relationship models to improve image retrieval performance.

Acknowledgments

This work would not be possible without the advice and assistance of Melanie Mitchell, my advisor on this thesis. Her suggestion to use [7] as the topic for my thesis was an excellent one, as I found it extremely interesting and packed with opportunities to learn more about machine learning.

I would also like to thank Justin Johnson for his kind assistance in getting the original source code and data files for this effort.

Finally, this work would never have come to pass without the support of my family: Bernadine (Bing Bong), Reginald (Shesshie), Desmond (ManMan), Cordelia (Bobo), and most of all my darling wife Maggie.

Contents

List of Figures

List of Tables

Dedicated to my wife and children...

Chapter 1

introduction

It has been said that a picture is worth a thousand words. While the origins of this phrase may be debated, the sentiment is clear: it is very challenging to generate concise descriptions of an image with text. Any two individuals attempting to fully describe a given image are likely to produce very different, yet valid, sequences of text. This process of converting an image to text remains difficult in the reverse direction as well. The difficulty of generating a representation for an image from a description is extreme. Simple descriptions can be filled with ambiguity, implied meaning, and vagueness.

Given these challenges, how can a machine learning system take a description and use it to retrieve images that match thexs description from a pool of images? At its core, retrieving images from a description requires some means of determining description-to-image similarity and then ranking the images by the similarity metric. The process of taking an image and determining its similarity to another representation remains challenging. If a picture is worth 1,000 words, does it require 1,000 words to describe an image? An unreasonable requirement for users, to be sure. If we are not willing to describe desired images with a highly detailed description, we must be willing to explore alternative representations and to accept some degree of imprecision in the results that are retrieved from the system.

In this thesis I investigate the image retrieval method proposed by Johnson et al. [7] which describes a framework using scene graphs to represent the detailed semantics of an image. In order to leverage the expressiveness of the scene graph representation, image retrieval queries are represented as scene graphs as well. The framework captures the detailed semantics (objects, attributes, and relationships) of a desired image with a scene graph. Scene graphs were chosen as the query representation format because they provide a means to explicitly model a set of objects, the attributes of the objects, and the relationships between the objects at a high level of detail. An example of an image with its corresponding scene graph is found in Figure 1. In the training phase, structured scene descriptions generated by Amazon Mechanical Turk (AMT) workers are used as the ground truth representations for training a pipeline of models. To perform inference, a scene graph is used to represent the query and a conditional random field (CRF) is generated from its structure. Each image in the candidate set is processed by the pipeline of models, producing a set of scores that are used by the CRF generated from the query. Belief propagation is used to perform inference on the configured CRF and the energy value of the solution is used as a measure of similarity for the candidate image to the scene graph query. The set of energy values generated from the candidate images is used to rank the images, producing an ordered arrangement of images for retrieval.

In this thesis, I propose two changes to the method of [7] in an effort to improve retrieval results. The proposed changes provide a means for the IRSG model to make judgments about object classes that it has not been trained to recognize. In [7] when these types of objects are found in the scene graph they (and their attributes and relationships) are not included in the energy calculation. I hypothesize that the changes will improve image retrieval performance by allowing the modified model to make use

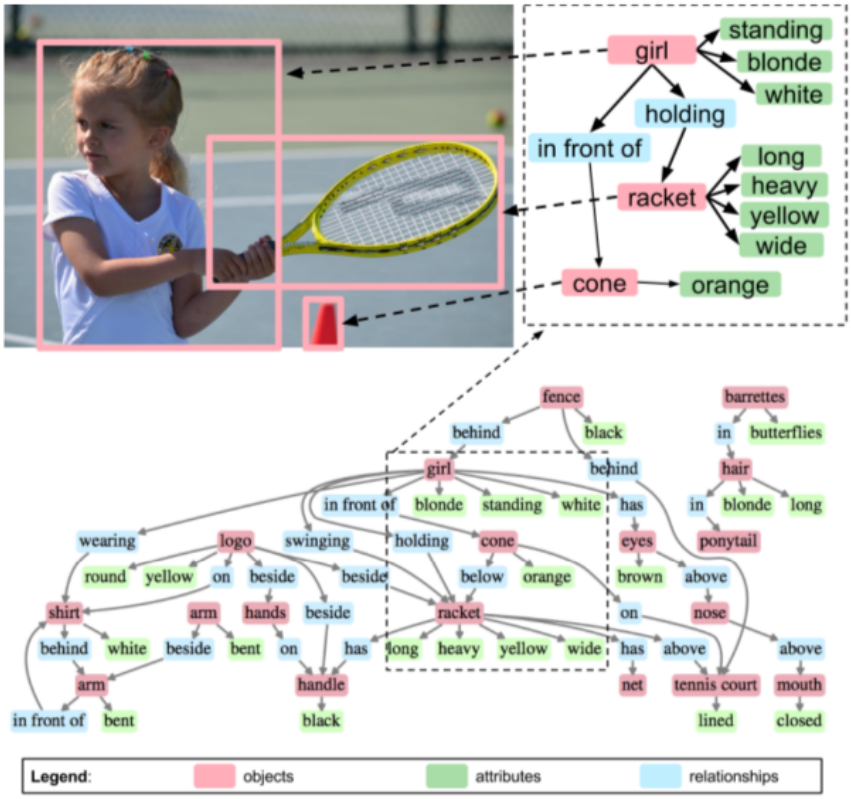


FIGURE 1: An image and its corresponding scene graph from [7]. A subset of the full scene graph depicts a *grounding* of the scene graph objects to specific bounding boxes from the image. The grounded scene graph depicts the location of the `girl`, `racket`, and `cone` objects within the image. Attribute groundings (e.g. `blonde`, `white`, `orange`) are co-located with their parent object bounding boxes. Also represented are the relationships between the objects: `girl holding racket` and `girl in front of cone`.

of semantic content that is discarded by the original model. In the following chapters, I will cover related work in scene graph based image search, explain the architecture of the original and modified model, detail the experiments I performed, make conclusions about the effectiveness of the changes, and finally discuss avenues for future effort in this area.

Chapter 2

related work

In this chapter I review some recent work related to the topic of semantic image retreival with scene graphs.

Image Representation. Recent efforts in generation of text descriptions from images often hinge on learning multimodal embeddings of the image features and sentence features, and then using the image features and a neural network with some manner of recurrent structure to determine the most likely next word [9, 20]. Johnson et al.'s IRSG uses a similarity metric as well, but it is by means of a probabilistic graphical model, and compares components of the model with a structured representation in the form of a scene graph, rather than a natural language sentence. The work by Mitchell et al. [15] takes a different approach to caption generation by analyzing a corpus of text to learn a probability distribution for the noun, adjective, and preposition co-occurences and computer vision detections to generate the objects and attributes. The detected objects and attributes serve as anchors in a tree that is grown into a descriptive sentence through the learned syntactic dependency priors and established sentence parsing guidelines. Evaluation of the generated captions uses Amazon Mechanical Turk (AMT) resources to score the captions on a 5-point scale over five different metrics.

Semantic Information from Images. Understanding the semantic information from images is an open problem. IRSG models the spatial relationships between object bounding boxes and uses the spatial model in conjunction with object scoring models to make judgments about similarity. Zitnick et al. [21] approach this problem by learning from artificial scenes generated by AMT workers. The scenes do not make any attempt to be photorealistic, but instead are generated from clip art figures and objects arranged to match a text description. The authors hypothesize that an image's semantic content can be learned without requiring a photorealistic level of detail in the source content. This use of clip art cartoons arranged by humans produces a data set with human-generated and human-annotated scenes. The data consists of 1,002 scene descriptions, with 10 human-generated scenes per description resulting in a total of 10,020 images in the data set.

After creating the data set, the objects are used to generate features from the objects in the images: object presence, object location, object attributes, relative position of objects, and what objects the people were holding and wearing. mutual information and conditional mutual information scores were calculated for the features to determine the relevance of objects and relationships to the scenes; these mutual information scores were then analyzed to determine which features were most relevant to the scenes. Finally, the features are used to study semantic similarity of the scenes, with the features used as values for a nearest-neighbor style image retrieval. Recall is used as the primary metric for quantitative measurement of the performance of the image retrieval task.

Chang et al. [3] use scene graphs for scene description, but the scene graphs are generated from text and used as an intermediary step in the process of text to 3D

scene generation. The primary focus is on learning spatial relationships between the objects and representing objects that are implicitly part of the scene but not explicitly mentioned in the description. The text scenes are converted to scene graphs by leveraging learned priors from a set of data. The priors inform the way that certain object types physically support other object types, which objects are present in certain scene types, and the typical physical arrangement of objects in a scene type. The scene graphs and learned features of objects and their relationships are used to generate scenes from a set of stock 3D models. All measures of performance for this effort are qualitative in nature. IRSG also uses its scene graphs as scene description, but the scene graphs are used for image retrieval rather than for generation of scenes. Chang et al. focus on learning spatial relationships and using them to achieve their goals, but use the spatial relationship data to place objects in a scene in a natural arrangement rather than scoring existing relationships.

NEIL. The Never Ending Image Learner [4] is a machine learning system that is engaged in a cycle of image retrieval and learning, intended to run continually without human intervention. It relies on Google Image Search for the retrieval phase, during which it requests images from its known classes of objects, attributes, and scenes. The labels of the objects, attributes, and relationships are drawn from the Never Ending Language Learner [2], a Natural Language Processing model that inspired NEIL. The retrieved images are clustered to remove outliers, and the remaining images are used to refine the detectors. Once the detectors have been trained, relationships are extracted from the images through co-occurrence analysis for object-object, object-attribute, scene-object, and scene-attribute relationships. NEIL uses its learned relationships during the image refining stage to ensure that the retrieved objects possess the expected characteristics (cars have wheels, pizzas are round, etc.), this helps avoid the semantic drift that can occur as the system retrieves and refines

its understanding of objects, scenes, and relationships.

Like IRSG, NEIL seeks to learn semantic information from images, but with the use of clustering for classification of the known types (objects, attributes, scenes) and a variant of linear discriminant analysis for generation of bounding boxes. In contrast, IRSG uses convolutional neural networks and Gaussian mixture models for scoring the objects and relationships, and a Probabilistic Graphical Model for determining how well the image matches a scene graph description.

Chapter 3

background

In this chapter I describe the details of the Image Retrieval using Scene Graphs (IRSG) model proposed and implemented by Johnson et al. in [7]. The model is composed of several components, depicted as a block diagram in Figure 2. Each of the individual components will be described in this chapter in order to provide a better understanding of how the parts contribute to the whole of the model.

The dataset used for the IRSG model consists of 5,000 images manually drawn from the YFCC100m [19] and Microsoft COCO [13] datasets. Each of the 5,000 images is annotated by Amazon Mechanical Turk (AMT) workers, using an open vocabulary rather than a pre-specified set of allowed terms. The annotations produced by the human workers are corrected and verified by another set of AMT workers. This ultimately results in each image having a detailed scene graph, as well as bounding box coordinates for each object and attribute. The full data set is split into a 4,000 image training set and 1,000 image test set. The following subsections describe the pipeline components in detail.

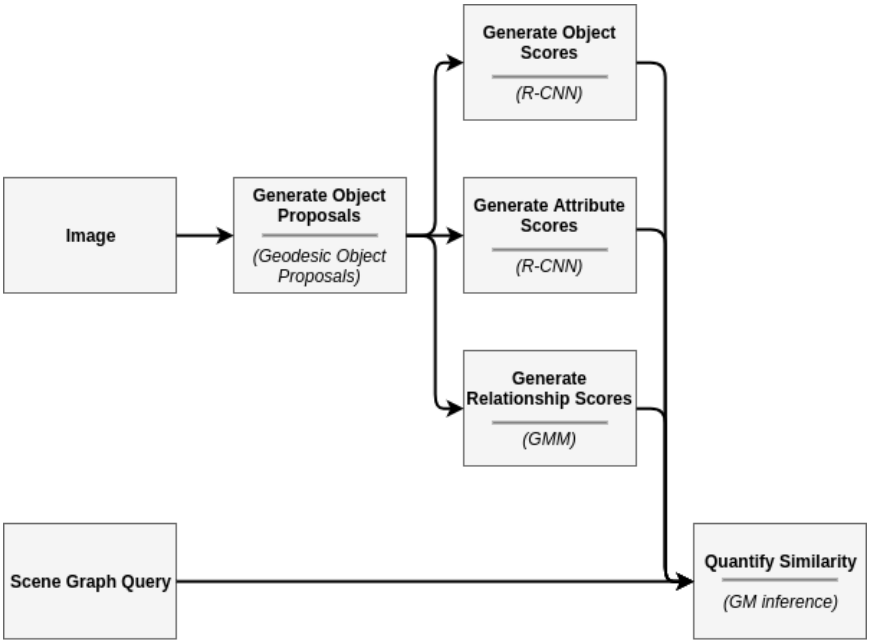


FIGURE 2: Block diagram of the image retrieval pipeline. This high-level diagram shows how the pipeline components process the input (image and scene graph) in sequence in order to generate a graphical model that can quantify (through inference) the similarity between the image and scene graph. The various components are described in detail in the text

3.1 GEODESIC OBJECT PROPOSALS

Every image needs its objects and attributes identified. In order to find candidate objects/attributes, IRSG first processes each image and generates bounding boxes around sufficiently interesting areas of the image. Each of these bounding boxes are presumed to contain an object of some type. The framework uses Geodesic Object Proposals [11] (GOP) to generate its candidate bounding boxes. A high-level depiction of the input and output for GOP is found in Figure 3

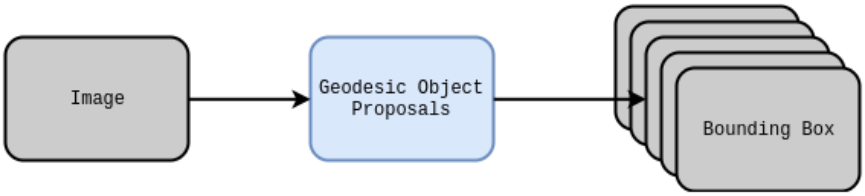
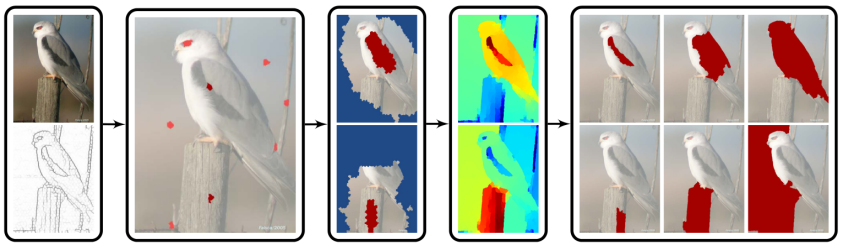


FIGURE 3: Input and output data for Geodesic Object Proposals. The image data is composed of the image pixels. The output is a set of candidate bounding box coordinates.

First, GOP oversegments the image into superpixel regions, where each superpixel is defined as a small cluster of image pixels with similar color. The superpixels are used as nodes in an undirected graph; each edge takes on the probability that the two nodes it connects span an object boundary found by edge detections. A configurable number of seed superpixels are selected with a linear ranking classifier trained to minimize an objective function. The objective function is designed to give a high rank to superpixels that are part of objects that have not yet had one of thier superpixels selected. The details of the dataset used for training are not described in [11].

After seed selection, a foreground set and background set of superpixels are generated for each seed, and the space between the foreground and background sets is progressively filled by level sets of a signed geodesic distance transform. These level sets are used to generate the object proposals. The number of object proposals is influenced by a parameter λ that determines how much signed geodesic distance should be accumulated in order to generate a new object. The pipeline from image to object proposals for a sample image is depicted in Figure 4. The geodesic object proposal method is configurable by the number of initial seeds and the granularity of the level set magnitude that defines a new object.



(a) Input (b) Seeds (c) Masks (d) SGDT (e) Proposals

FIGURE 4: Geodesic Object Proposal pipeline from [11]. The input phase (a) depicts the edge detection used during the construction of the superpixel graph. Once segmented into superpixel regions, seed superpixels are selected (b), and foreground/background masks are generated (c). The SGDT phase (d) calculates the Signed Geodesic Distance between the foreground and background regions. These distance values are then used in the proposal stage (e) to generate the areas corresponding to the proposed objects. Each proposal object is surrounded by a bounding box; these boxes are the ultimate output of the method.

3.2 CONVOLUTIONAL NEURAL NETWORKS

Convolutional Neural Networks (CNN) are used in IRSG to generate object and attribute class scores for each bounding box. These scores indicate how likely it is that the box contains an item of the known object and attribute classes and attribute classes. A high-level depiction of the input and output for the CNN model, called R-CNN is shown in Figure 5.

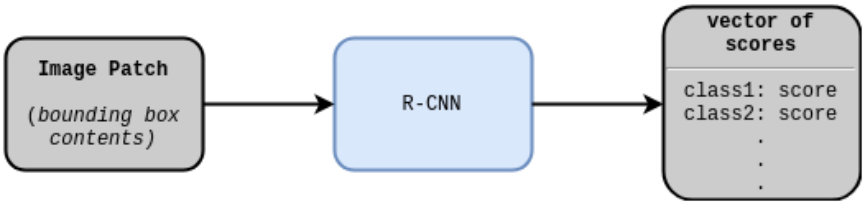


FIGURE 5: Input and output data for the convolutional neural network (R-CNN). The input image patches are generated from the bounding box coordinates provided by Geodesic Object Proposals. The output is a vector of scores for the bounding box, one element for each object class recognized by the CNN.

CNNs have been used as whole-image classifiers with great success, but only recently have they been able to be effectively applied towards object classification within an

image. A CNN works by a progressive conversion of high dimensional data through a sequence of convolutional, pooling, and nonlinear transformations. The network's convolutional filters are learned through training to maximize classification performance by backpropogation of error. The pooling layers reduce the dimensionality of the data without the addition of parameters that would be incurred from additional convolutional layers. This progressive reduction in dimensionality through convolutional filtering and pooling leads to each layer identifying higher level features. The final layer is typically a fully connected layer that maps a vectorized representation of the input image onto a set of potential classes.

IRSG uses a CNN that is configured to generate scores for the contents of the bounding boxes generated by the Geodesic Object Proposal model, rather than a single classification for the entire image. This architecture is described in Girshick et al. [5] and is referred to as "Regions with CNN features," or R-CNN.

To train the scoring system used for objects and attributes in IRSG, the authors mined the training data to determine which object and attribute classes occur frequently enough to warrant scoring. Any object or attribute with 50 or more instances would be included in the set of classes to be recognized by the R-CNN. The authors train two CNNs for this task: one for object scoring, one for attribute scoring. Each CNN is pre-trained on the ImageNet dataset [18], then fine-tuned on the object and attribute classes identified in the training set. The bounding boxes generated by the object proposal method are run through both of the R-CNN models when performing inference on an image for retrieval. The resulting scores are used further down in the pipeline when determining how well the image matches the query. The end result of this process is that each of the B bounding boxes in the image produces a vector of dimension $|\mathcal{C}|$ where $\mathcal{C}$ is the set of classes to be recognized (object classes or attribute

classes).

3.3 GAUSSIAN MIXTURE MODELS

Gaussian Mixture Models (GMMs) are used in IRSG to model the spatial relationships between bounding boxes in the image. A high-level depiction of the input and output for this model is shown in Figure 6.

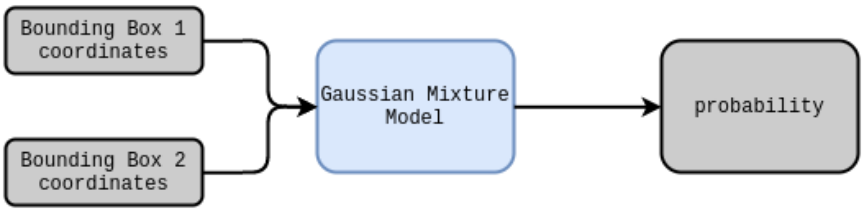


FIGURE 6: Input and output data for the Gaussian mixture model. The input is a pair of bounding box coordinates. The output is the probability that a relationship between the input bounding boxes could be drawn from the data learned by the Gaussian mixture model.

Given any two object bounding boxes, the x, y, width, and height values of the two boxes are converted to a vector that express their relative position and scale. This conversion process is depicted in Figure 7.

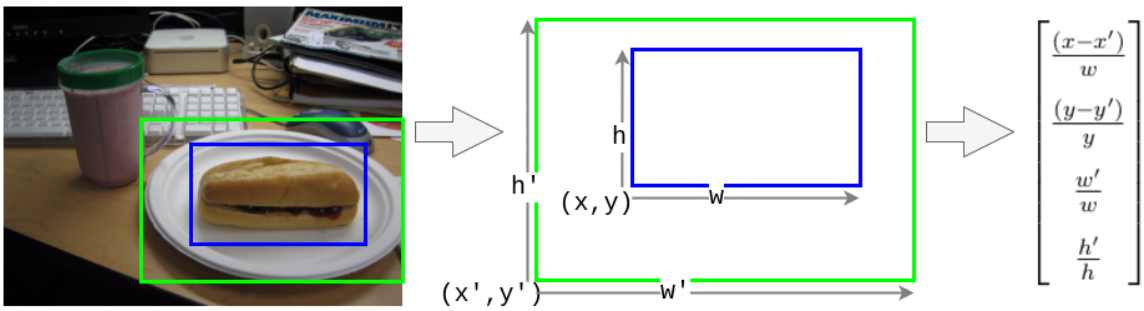


FIGURE 7: Converting two bounding boxes for the relationship "sandwich on plate" to a single vector. The x and y values are the pixel locations of the lower-left corner of each bounding box. w and h are the width and height (in pixels) of the bounding box.

The GMM is a model of the form [16]:

$$p(\mathbf{x}|\theta) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}|\mu_k, \mathbf{\Sigma}_k)$$

where $\mathbf{x}$ is the relative position and scale vector generated from the two bounding boxes and θ represents the learned parameters: mean (μ_k), covariance ($\mathbf{\Sigma}_k$), and mixing parameter (π_k) for each of the K components. The model sums K multivariate normal (MVN) distributions to arrive at the probability that the vector $\mathbf{x}$ would arise from the parameters θ. The value of K is specified prior to training and can be selected through a parameter selection method such as cross-validation.

In IRSG, the GMM is used to calculate the probability that any two bounding boxes represent the relationships learned from the training data. Much like for the objects and attributes, the authors mined the training data for commonly occurring relationships. There were two types of relationships modeled: object-explicit and object-agnostic. The object-explicit GMMs model the most frequently occurring object-relationship-object configurations, e.g. `man wearing pants`, `clock on wall`, `mouse next to keyboard`, etc. The object-agnostic relationships model any objects in the specified relationship e.g. `* wearing *`, `* on *`, `* next to *`. The object-explicit models are more accurate, but require specific objects to be in a specific order. For instance, the `mouse next to keyboard` GMM cannot be used to model a `keyboard next to mouse` relationship because the order of the bounding boxes matters in this model. The object-agnostic relationships are more general-purpose and can be used when there is a relationship in the scene graph that does not have an object-explicit model, but is a common enough relationship to warrant a scoring model.

Probabilistic graphical models (PGMs) are a powerful framework that use a graph-based representation for compactly encoding a complex probability distribution over a high-dimensional space [10]. PGMs employ a flexible system of notation that allow them to model a wide variety of dependence and independence configurations. IRSG uses a factor graph representation [12] to model the objects, attributes, and relationships expressed in the scene graph query. A high-level diagram of the input and output for this model is found in Figure 8.

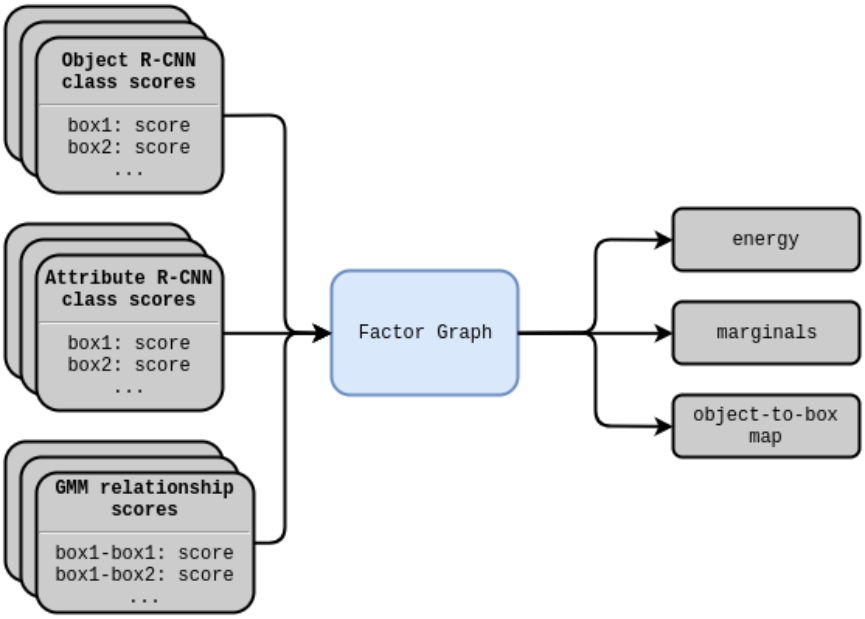


FIGURE 8: Input and output data for the graphical model. The three inputs are: bounding box to score mappings for each object class recognized by the object R-CNN model, bounding box to score mappings for each attribute class recognized by the attribute R-CNN, and bounding box pair to score mappings for each relationship recognized by the GMM. The three outputs are: energy, marginal probability scores for each scene graph object, and scene graph object to bounding box mapping

A factor graph is a bipartite graph composed of an "object node" for each scene graph

object (`man`, `car`, `bench`), and a "function node" for each set of scores from the R-CNN or GMM. The object nodes, depicted as white circles, are connected only to the function nodes, depicted as black squares. An example factor graph generated from a scene graph is shown in Figure 9. Function nodes from the R-CNNs are referred to as "unary potentials" because they affect just one object node, Function nodes from the GMM are referred to as "binary potentials" because they impact the two object nodes that constitute the relationship.

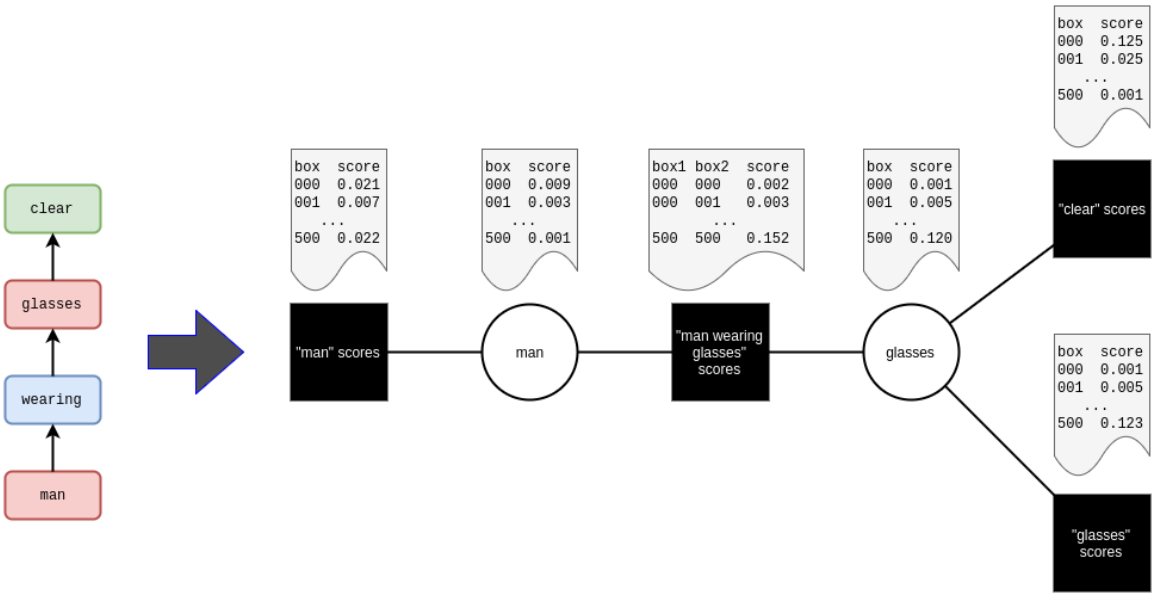


FIGURE 9: A factor graph generated from the scene graph query "man wearing clear glasses." Assume for the sake of this example that an image with 500 bounding boxes is used to generate the scores for this factor graph. The scores for each of the 500 bounding boxes from the R-CNN object model, R-CNN attribute model, and GMM are depicted above the corresponding function node. In practice, these scores vary for every image. The R-CNN object model provides scores for the `man` and `glasses` function nodes. The R-CNN attribute model provides scores for the `clear` function node. The GMM provides scores for the `man wearing glasses` function node.

The OpenGM [1] Python library is used in [**?**] to generate factor graphs. The factor graph is generated by successively scanning the scene graph and adding structure from object nodes, to unary functions, to binary functions. The first step is

to add object nodes, one for each object in the scene graph query (e.g. `man` and `glasses`). Next, each bounding box is scored by the object R-CNN and the score for the appropriate class is loaded into its function node. In the example from Figure 9, all 500 bounding boxes are scored by the object R-CNN. The `man` class scores for each bounding box are loaded into the `man` unary function node, and the `glasses` class scores for each bounding box are loaded into the `glasses` unary function node.

The final step in configuring the factor graph is to load the binary function nodes with scores from the GMM. Each pair of bounding boxes is processed by the GMM for the specified relationship (`man wearing glasses` or `* wearing *`), resulting in 500^2 scores that are loaded into the `man wearing glasses` binary function node.

Now that the object nodes, unary functions, and binary functions are generated and loaded with scores, performing inference on the graphical model will result in the object node placeholder values taking on the values that maximize the bounding box to scene graph object mapping. An example of the conversion of scene graph to factor graph is depicted in Figure 10. I cover the details of the objective function in the Section 3.5.

The OpenGM library provides numerous algorithms for approximate and exact inference. IRSG uses the loopy belief propagation (LBP) approximate inference algorithm. LBP is used because, in practice, it often reaches the same result as exact inference methods but has a run time orders of magnitude faster than exact methods [8]. For a factor graph with N object nodes and B bounding boxes, there are B^N possible configurations, each configuration having a different energy value. With the average graph having 14 objects and the average image having 632 bounding boxes, we are tasked with finding the lowest energy configuration among these $632^{14} \approx 1.6 \times 10^{39}$

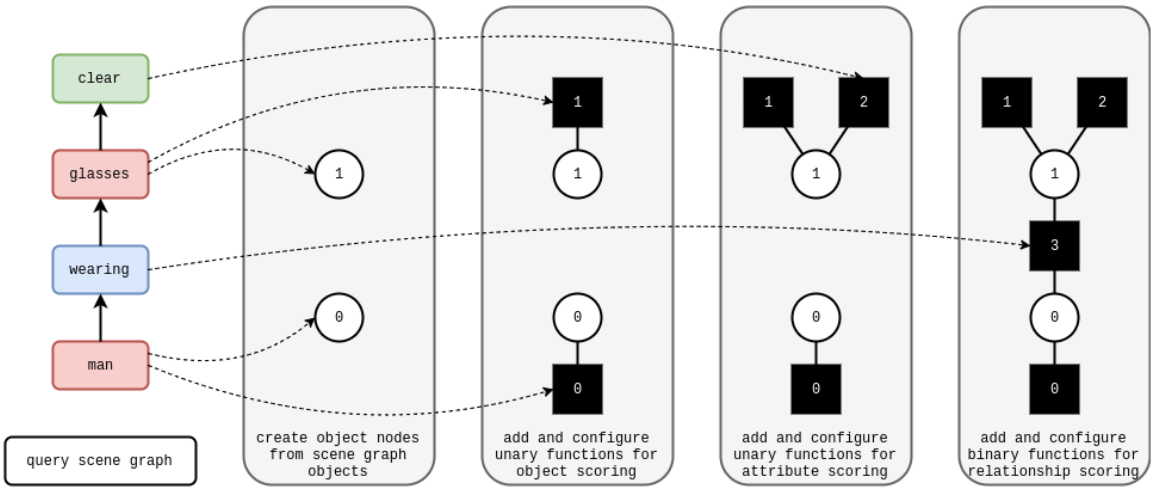


FIGURE 10: Converting a scene graph to a factor graph. The scene graph is progressively altered to pull more information into the scene graph.

possible configurations. I tested a brute force exact inference in order to compare the exact and approximate inference methods. Runtime for exact inference was over six hours and generated the same results as LBP, which completed in less than a second. After completing inference, the PGM returns an energy value that quantifies the similarity of the factor graph generated from the scene graph to the arrangement of objects, attributes, and relationships found in the image. This energy value is used by IRSG to rank all the test images as a function of their similarity to the query scene graph, where a low energy value indicates a high degree of similarity, and a high energy value indicates low degree of similarity.

3.5 IMAGE RETRIEVAL MODEL

As previously described, IRSG is composed of several models arranged in a pipeline. The pipeline takes an image and a scene graph query as input and produces three outputs: an energy value indicating the degree of similarity between the image and scene graph, a "grounding" consisting of a mapping of the scene graph objects to

image bounding boxes, and the marginal distributions for each object in the scene graph. The grounding generated by IRSG corresponds to the configuration of bounding boxes that give rise to the lowest energy configuration of the factor graph. The components of the pipeline are briefly described in Table 1.

Component	Purpose
Geodesic Object Proposal	Generation of bounding boxes
R-CNN	Object scores of bounding boxes
R-CNN	Attribute scores of bounding boxes
Gaussian Mixture Model	Relationship scores of bounding box pairs
Factor Graph	Similarity calculation

Table 1: IRSG component descriptions

Notation. The images in the data set used for training the model components are all annotated with detailed scene graphs, and these graphs are used to generate the set of object classes $\mathcal{C}$, attribute types $\mathcal{A}$, and relationship types $\mathcal{R}$. The scene graphs used to describe the images and queries are composed of objects, attributes, and relationship nodes. Each scene graph consists of a graph G with objects O and edges E. Each scene graph object is composed of an object class from $\mathcal{C}$ and the set of attributes for that particular instance of the object $o_i = (c_i, A_i)$. This allows for multiple instances of the same class of object in the scene graph e.g., `white car`, `black car`, and `blue car`. When objects are connected by an edge, that edge expresses a relationship (`on`, `next to`, `wearing`) between the two objects. An example of a scene graph with 3 objects, 2 relationships, and 2 attributes is shown in Figure 11.

The geodesic object proposal method will generate a set of bounding boxes B, and the process of mapping scene graph objects $o_i = (c_i, A_i)$ to bounding boxes will be a key element of the image retrieval process. The full mapping of objects to bounding boxes $\gamma : O \to B$ is composed of a mapping for each object $o \in O$ such that γ_o represents the mapping of object o to a specific bounding box.

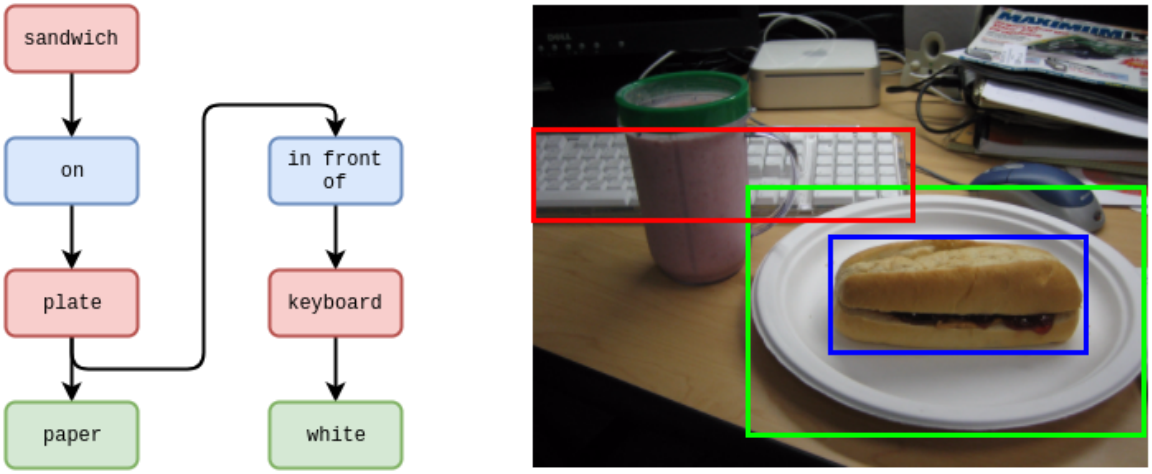


FIGURE 11: A scene graph (left) corresponding to an image (right). The objects are a `sandwich`, `plate`, and `keyboard`. The `plate` has the attribute `paper`, and the `keyboard` has the attribute `white`. The `sandwich` and `plate` are connected with the `on` relationship edge, and the `plate` and `keyboard` are connected with the `in front of` relationship.

Model. As briefly described in Section 3.4, IRSG seeks a mapping γ with maximum probability conditioned on the scene graph G and set of bounding boxes B:

$$P(\gamma|G,B) = \prod_{o \in O} P(\gamma_o|o) \prod_{(o,r,o') \in E} P(\gamma_o, \gamma_{o'}|o,r,o')$$

Since the system does not have models that express $P(\gamma_o|o)$, Bayes' rule can be applied to change $P(\gamma_o|o)$ to $P(o|\gamma_o)P(\gamma_o)/P(o)$. There are a vast number of possible combinations for a given o, as there are 266 possible object classes and 145 attribute classes. We can expect $266 \times \sum_{k=0}^{145} \binom{145}{k} = 266 \times 2^{145} \approx 1.2 \times 10^{46}$ possible configurations for o, where many arrangements are just as likely as any other. Furthermore, there is not enough data to reasonably construct an appropriate prior for this distribution. Given this, we will assume an uninformative prior for $P(o)$, and for similar reasons I assume an uninformative prior in the form of a uniform distribution for $P(\gamma_o)$. With uniform priors for $P(\gamma_o)$ and $P(o)$, these terms become constant and can be dropped from consideration when performing inference. After application of

Bayes' rule and dropping the constants, we have:

$$P(\gamma|G,B) = \prod_{o \in O} P(o|\gamma_o) \prod_{(o,r,o') \in E} P(\gamma_o, \gamma_{o'}|o,r,o') \tag{3.1}$$

Although $\prod_{o \in O} P(o|\gamma_o)$ cannot be directly modeled, we can take advantage of the fact that each $o \in O$ is a combination of an object class c and a set of attributes $o = (c, A)$, so $P(o|\gamma_o)$ can be decomposed. It can be split into the object class probability $P(c|\gamma_o)$ and the attribute probabilities $\prod_{a \in A} P(a|\gamma_o)$. These probabilities are the unary functions in the factor graph, and are extracted from the object and attribute R-CNN scores, respectively. The class scores for each R-CNN are scaled with a Platt model [17] to create probability estimates. Similarly, the GMM outputs are scaled with a Platt model so that the probabilities generated by the various relationship models can be more fairly compared during inference.

The binary function that models spatial relationships between bounding boxes is discussed in the GMM section 3.3. For this function, the bounding box pair $\gamma_o, \gamma_{o'}$ is converted to a feature vector for scoring by the GMM. As discussed in the GMM section, if there is a model for the specific object-relationship-object (o, r, o') tuple e.g. `man wearing glasses`, the object-explicit GMM will be used. In the case where there is no object-explicit model available, the object-agnostic model $P(\gamma_o, \gamma_{o'}|r)$ is used.

With this formulation for the probability of the object-to-box mapping conditioned on the scene graph and bounding boxes, IRSG seeks to maximize the expression in Equation 3.1 through Maximum a Posteriori (MAP) inference on the factor graph generated from the query. With the aim of maximizing γ, the objective of the learning

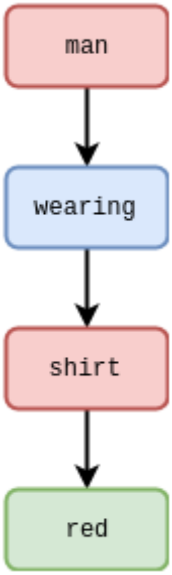
process becomes:

$$\gamma^* = \arg\max_{\gamma} P(c|\gamma_o) \prod_{a \in \mathcal{A}} P(a|\gamma_o) \prod_{(o,r,o') \in E} P(\gamma_o, \gamma_{o'}|o, r, o') \tag{3.2}$$

This final expression is modeled with the factor graph, using the object and attribute scores from the R-CNNs for the unary factors, and the Gaussian mixture model scores for the binary factors. Inference on the factor graph provides the set of bounding boxes which maximize the product of the unary and binary potentials.

3.6 QUALITATIVE OBSERVATIONS

This section is intended to show how IRSG scores objects in an image and how those scores shift during inference to "lock on" to the appropriate image objects given the semantic details in the scene graph query. The following images test the query `man wearing red shirt`. The original image and the scene graph query are shown in Figure 12.



(A) scene graph query

(B) query result

FIGURE 12: A small query and an image from the test set. This query performs particularly well on this image, as seen in the bounding boxes being fairly well-localized on the expected objects.

First, we look at how the `man` object is scored by the object R-CNN. Figure 13 shows a heat map of the scores for each of the 315 bounding boxes generated in this images by the geodesic object proposals. For each pixel of the heat map, the `man` class score for any of the 315 bounding boxes that cover that pixel is summed up. After generating the pixel values, a slight Gaussian blur ($\sigma = 7$) has been added to improve visual quality of the heat map. The heat map data is rendered slightly transparent and overlaid on the original image for a visual reference. Figure 14 shows object R-CNN scores for the `shirt` object class.

FIGURE 13: A heat map of the object R-CNN values for the scene graph object `man`. The scores are somewhat diffuse about the man and woman, as the R-CNN does not excel at distinguishing whether the figure is a man or woman from just the image data.

FIGURE 14: A heatmap of the object R-CNN values for the scene graph object `shirt`. The high scoring areas are centered on the shirts of the man and the woman, because the scores are focused solely on *shirt-ness* and do not take the wearer of the shirt into account.

After performing inference on the factor graph, the marginal probability values for the `man` object node and `shirt` object node can be displayed in a similar fashion as the R-CNN scores for these objects. The following images demonstrate visually how IRSG uses semantic information to narrow the field of candidate bounding boxes by leveraging the attribute and relationship scores and the scene graph structure.

Figure 15 shows the post-inference scores for the `shirt` object. The methodology for generation of these heat maps is identical to that used for the R-CNN scores, although the data source is the object node marginal probability values from the factor graph, rather than the R-CNN class scores. The factor graph is able to use the `red` attribute scores to reduce score values over the woman's blue shirt and increase scores over the man's red shirt. This leads to an increase in the score values for bounding boxes that lie on the man's red shirt. Figure 16 shows the post-inference scores for the `man` object. Note how the scores are more tightly focused on the man in the image, whereas the initial scores were more diffuse about the man and woman. IRSG has used the semantic information about the `man` object wearing the red shirt to improve the scoring of bounding boxes that lie over the man in the image, while reducing the scores for bounding boxes that lie over the woman in the blue shirt.

FIGURE 15: A heatmap of the factor graph object node scores for the scene graph object `shirt`. As compared to the R-CNN scores for the shirt, they are more focused on the man's shirt, since it is red. The attribute scores help draw scores higher for the man's red shirt and away from the woman's blue shirt.

FIGURE 16: A heatmap of the factor graph object node scores for the scene graph object `man`. The scores are more localized over the figure of the man in the image as compared to the R-CNN scores. IRSG has successfully used the semantic information about the shirt to better localize the man in the image.

These images provide a nice visual representation of how the IRSG model can leverage learned relationship and attribute scoring to infer semantic information that can assist in the context of an image retrieval task. The initial scores for the `man` object were fairly diffuse about the two human figures; after inference, the object nodes were able to infer that the figure on the right was more likely to be the `man` because the figure on the right is wearing the red shirt while the figure on the left is wearing a blue shirt. Similarly, inference allows the IRSG model to identify the red shirt as the one on the right side of the two shirts in the image.

Chapter 4

my experiments and results

In IRSG as described in [7], the scene graphs are created using an open vocabulary. There are 6,745 object classes, 3,743 attribute classes, and 1,310 relationship classes in the full dataset. The IRSG model was trained using a subset of these classes: 266 object classes, 145 attribute classes, and 68 relationship classes. These classes are selected from the most frequently occurring classes. This results in situations where IRSG is presented with a scene graph that contains classes that it has not been trained to recognize; these classes are *unrecognized* by the scoring models.

As an example, Figure 17 shows a scene graph containing `black toque on man`. IRSG has not been trained to recognize the object class `toque`, so it is unrecognized. In [7], unrecognized scene graph objects are not entered into the factor graph, so any recognized attributes (`black`) and relationships (`...on man`) that are associated with the unrecognized object are also not taken into account during the factor graph similarity calculation. If the intent of the model is to retrieve images by leveraging learned semantic content, it would be advantageous to use unrecognized class scores and relationship scores in the calculation of similarity values. In this chapter I describe the experiments I carried out to address this issue. In particular, I extended the methodology of [7] to provide scores for unrecognized objects so that they can be

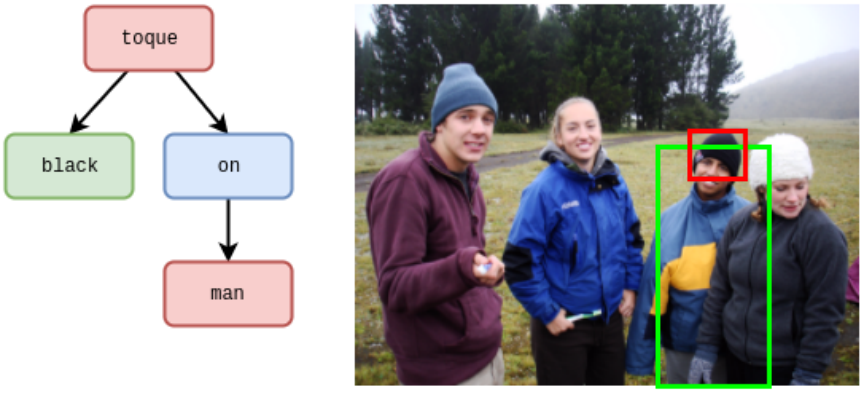


FIGURE 17: A scene graph with an *unrecognized* object, `toque`, in the red bounding box. For the original method, the attribute `black` and relationship `on` will not be added to the factor graph when determining similarity of this image to the scene graph. This results in a factor graph that employs only the object scores for the object class `man`, and does not take the `black` object on the `man` object into account when calculating the similarity value. Best viewed in color.

added to the factor graph. With the unrecognized object added to the factor graph, the associated attribute function nodes and relationship function nodes can be added as well. I extended the model only for unrecognized objects; any unrecognized attributes or relationships will be ignored and not added to the factor graph.

Uniform Scoring Model. The first method I propose for scoring unrecognized objects is to use a uniform distribution of scores. This method assumes that the probability of the unrecognized object being in a bounding box is the same across all of the image's bounding boxes. Given an image with the set of bounding boxes B and a scene graph with an unrecognized object class $\widetilde{c}$, the uniform scoring model is defined as $P(\widetilde{c}|\gamma_o) = \frac{1}{|B|}$. The uniform scoring model is used in the place of $P(c|\gamma_o)$ in Equation 3.2. Although the uniform scoring model is uninformative with respect to the object in question, I anticipate retrieval performance to improve. The original IRSG model discards the associated attribute scores and relationship scores that depend on the unrecognized object, whereas now the factor graph can apply these associated scores to the inference solution.

Empirical Scoring Model. The second method I propose for scoring unrecognized objects operates under the assumption that the system is more likely to find objects in regions with relatively more objects than it would in areas that are relatively devoid of objects. When used to score a bounding box for the presence of an unrecognized scene graph object, the empirical scoring model averages the scores for all *recognized* scene graph objects in the image with respect to the bounding box in question. To elaborate, when a bounding box is presented to the object R-CNN, it returns a vector of 266 scores, one for each object class. The empirical scoring model will take the average of the scores for the recognized scene graph objects and use that value as the score for the unrecognized object. So, when `man` and `shirt` are recognized objects in the query graph, the model returns the average score for those classes for any undetected object in the scene graph.

As in the uniform model, the empirical scoring model generates scores for unrecognized scene graph objects $P(\widetilde{c}|\gamma_o)$. Rather than assuming equal scores for all boxes it calculates, for each box, the average score for of all recognized object classes C_r in the scene graph:

$$P(\widetilde{c}|\gamma_o) = \frac{\sum_{c \in C_r} P(c|\gamma_o)}{|C_r|}$$

Unlike the uniform scoring model, this model assumes a nonuniform distribution of scores among the bounding boxes. This should result in a distribution of scores more tightly clustered in regions of greater recognized object density. As in the uniform model, associated attributes and relationship scores can now be added to the factor graph, so I expect an increase in retrieval performance.

Python IRSG Implementation. It is important to cover some details in regard

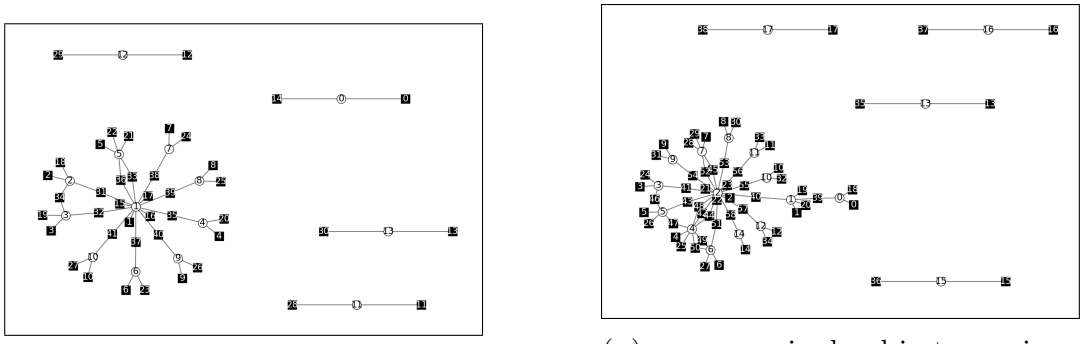
to the model implementation. I reached out to the primary author for any existing code and data, and they assisted in providing source code and data files. The existing source code was in MATLAB, and in an effort to learn in detail how the model works and to make the model more accessible, I converted the MATLAB version to Python[1]. The data files included with the MATLAB code did not contain R-CNN weights for the object and attribute scoring models; rather, one of the data files contained all 266 object class scores for each of the 5,000 image's various bounding boxes. The file also contained a similar structure with the attribute scores for each image's bounding boxes.

My initial experiments were to ensure that the Python version performed identically to the original MATLAB version. After the Python version was functional, I compared the γ mappings generated by the two versions to ensure that they were identical. This was a straightforward process of initially verifying that the energy values were equal for arbitrarily chosen image/scene graph pairs. This process involved verification that the bounding boxes assigned to the factor graph variables were identical between the two versions. Once manual verification of arbitrarily selected pairs were matching, I performed more thorough verification of all images against a single query. Energy values agreed nearly identically ($\pm 1 \times 10^{-5}$) between the two versions and bounding box selection was identical for all images given the verified queries. This validated that my Python version was performing identically to the MATLAB version. After ensuring that the Python version was performing as expected, I implemented my proposed scoring models for unrecognized objects, using the Python code base and adding a configuration variable to allow easy selection of whether to run the image retrieval by the original method, with the uniform detector, or with the empirical

[1] https://github.com/econser/py_irsg

method.

Figure 18 demonstrates the difference in the factor graphs arising from a scene graph with undetected objects. The factor graph on the left side is from the original IRSG, with 12 object nodes and 40 function nodes. The right side shows a factor graph generated from the same query, but using the unrecognized scoring model to allow the associated attributes and relationship functions to be added to the graph. This factor graph contains 17 object nodes and 58 function nodes, demonstrating that the unrecognized scoring models are correctly adding the score nodes that the original method was discarding.



(A) original method

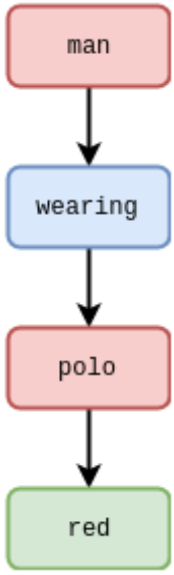
(B) unrecognized object scoring method

FIGURE 18: Graphical models built from the same graph but with different methods

Qualitative Observations. With the Python version working properly, I tested the Unrecognized Object Scoring models to validate that they were functioning as expected. Referring back to the qualitative observations in Section 3.6, I ran a similar scene graph query with the same image (Figure 12b) to observe the behavior of the revised model. Because both objects in the query (`man` and `shirt`) are recognized, the scene graph query had to be modified to ensure that one of the objects was unrecognized. I chose to change the recognized object class `shirt` to the unrecognized

object class `polo`, as in a polo shirt. Changing the `shirt` object allows the attribute R-CNN scores to play into the factor graph inference, hopefully demonstrating that these scoring models provide a way for the IRSG model to operate on unrecognized query objects.

First, the performance of the Uniform Scoring Model is presented.



(A) scene graph query

(B) query result

FIGURE 19: A small query that has been modified to trigger the use of an unrecognized scoring model. In this case, the uniform scoring model is used. This query performs well on the image, as seen in the bounding boxes being fairly well-localized on the expected objects. It is difficult to see the slight difference in the `shirt` bounding box as compared to the original method, and the `man` bounding box is identical to the original.

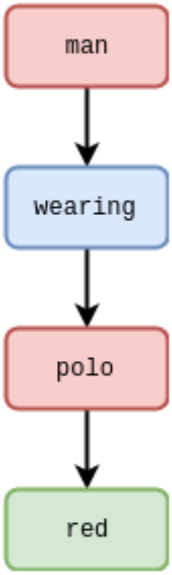
The R-CNN scores for the recognized object `man` are identical to what was presented in Section 3.6, so only the scores for the unrecognized object will be presented in this section. I anticipate the unrecognized object `polo` to be identified by the modified IRSG model, as the `red` attribute scores and the `wearing` relationship scores will factor into the process of inference. This is indeed the case for the Uniform Scoring

Model. It allows the IRSG model to process a query containing unrecognized objects, and properly locate the intended object. This modification allows IRSG to infer the presence of unrecognized objects in the context of an image retrieval task. Figure 20 shows how the uniform scoring model does not localize to the man's shirt, but after inference, the scores in Figure 21 show strong improvement in finding the red polo worn by the man.

FIGURE 20: A heatmap of the Uniform Scoring Model for the unrecognized `polo` object class. Although the scores are spread across the image, it is not completely uniform because bounding box overlap occurs more frequently in certain areas of the image and every box has the same score value. The man's shirt, where we would want high score boxes, is not particularly high scoring at this point.

FIGURE 21: A heatmap of the factor graph object node scores for the object `polo`. Here, the high scoring area of the image is nicely centered on the man's shirt, as anticipated. The factor graph is able to overcome the uninformative scoring of the Uniform Scoring Model by using semantic content in the image: the red object is the polo, and the man is wearing it. Note that this is a particularly well-suited image for this query, as there is only one large red object.

I now turn to the Empirical Scoring Model to determine if it is functioning appropriately. Using the same query and image, I observe similar results to the uniform method, with this Empirical Scoring Model producing a diffuse high scoring region as compared to the Uniform method, but more spread out than the original method. The energy and bounding box selection is remarkably similar to the original method. This indicates that the empirical method is operating as expected and is able to provide a means for the IRSG model to generate good results even when unrecognized objects are in the provided query.



(A) scene graph query

(B) query result

FIGURE 22: A small query that has been modified to trigger the use of an unrecognized scoring model. In this case, the empirical scoring model is used. The query performs well on this image, with the bounding boxes identical to those generated by the original method.

FIGURE 23: A heatmap of the Empirical Scoring Model for the unrecognized `polo` object class. The scores are not as widely spread across the image as seen in the Uniform method, but still exhibit a more widely spread high score area as compared to the original in Figure 14.

FIGURE 24: A heatmap of the factor graph object node scores for the object `polo`. Here, the high scoring area of the image is nicely centered on the man's shirt, as anticipated.

4.1 FULL SCENE GRAPH RETRIEVAL

In the qualitative observation sections above, I considered the performance of IRSG on a small scene graphs with two objects, one attribute, and one relationship. I now compare the performance of the empirical and uniform scoring models on the same Full Scene Graph retrieval task described in Section 6.1 of [7]. This retrieval task uses the full human-generated scenes, where the average scene graph contains 13.8 objects, 18.9 attributes, and 21.9 relationships. In this version of retrieval there can be unrecognized objects, as the model is trained on only the most common objects, and there is an open vocabulary. Because of this, I expect that results when using unrecognized scoring models will differ from the original method. In the Full Scene Graph task, unrecognized objects are not expected to be a common occurrence, because the authors train the original IRSG pipeline on the most frequent object, attribute, and relationship classes.

I executed the first 500 full scene graphs against all 1000 images in the test set. Only 500 were used due to the time required to run a full scene graph against the 1000 test set images, each batch of 1000 images requiring approximately 1 hour to execute. The experiments in this and the following sections were executed on Portland State University Linux lab machines: 8 core, 3.50GHz Intel Xeon CPU E3-1241 v3, 16 GB RAM. Each full scene graph query generates an energy value for each image in the test set, and the images are sorted in ascending energy level to determine the best-to-worst matching order.

Following [6] and [7], Recall at k (R@k) and Median Rank are the measures of performance I used for this and the following quantitative sections. Recall at k is calculated by determining the recall ($R = \frac{TP}{TP+FN}$) over a progressively increasing

number of ranked images, k. In the context of full scene image retrieval, TP (true positive count) is 1 when the one (and only one) correct image has been retrieved in the top k results, and 0 otherwise; FN (false negative count) is 1 when the image has not yet been retrieved in the top k results, and is 0 otherwise. The R@k value is the fraction of times that the correct image appears in the top k results for all n queries. Median rank is the median value of the retrieval rank for the n queries.

An R@k plot for the full scene graph queries up to $k = 100$ is shown in Figure 25. Recall values for $k > 100$ steadily increase to R=1.0, but offer little useful information. I choose to focus on recall performance for the top 10% of returned results, as a user of an image retrieval system is unlikely to scan much further than 100 results in an effort to retrieve a good result. Selected R@k result values are given in Table 2. The R@k

	Original	Empirical	Uniform
Median r	**14**	15	17
Average r	49.440	**48.704**	54.122
Std Dev r	95.299	**90.089**	100.214
R@1	**0.124**	0.118	**0.124**
R@5	**0.316**	0.306	0.296
R@10	**0.450**	0.434	0.426
R@55	0.754	**0.760**	0.740
R@60	0.776	**0.788**	0.754
R@65	0.790	**0.796**	0.774

TABLE 2: Median rank and R@k values for the full scene graph retrieval task. The three methods are very similar, with the original method slightly better in the low k range, and the empirical method slightly better in the mid-k range.

results show no significant difference in the recall performance of the three methods. The original method performs best at low values of k by approximately 1%, where performance matters most. The empirical method does show some improvement in the $50 \leq k \leq 70$ regime by the same 1% margin. The uniform method underperforms across all k values but $k = 1$. Median rank values are similar for the original

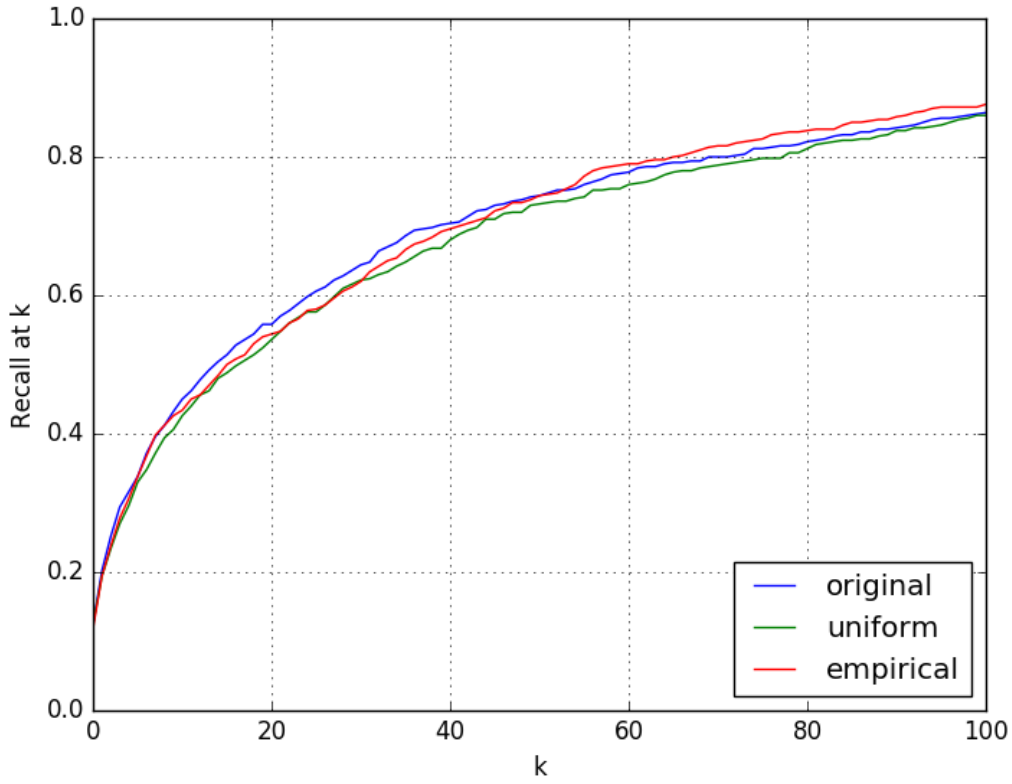


FIGURE 25: Recall at k for the full scene graph retrieval task. The three methods perform very similarly for this task, most likely because unrecognized objects are an uncommon occurrence in the full scene domain. As k increases, the recall value is the percentage of the 500 queries that have succeed in ranking the one correct image at or under the k^{th} rank.

and emprical methods, with the uniform method underperforming the other methods. It is interesting that R@k performance for the unrecognized scoring methods can negatively impact performance. These models only activate when unrecognized classes appear in the scene graph, so the only way that these models could reduce performance is when there is a scene graph or image configuration with unrecognized object classes that distracts the inference process from the expected object rather than focuses probability on it. Identifying these poor-performing configurations will provide a means of improving these scoring models.

The Partial Scene Graphs consist of 150 scene graphs generated by the authors of [7] from the training data. These small queries consist of two objects with one or zero attributes and one relationship, e.g. `sitting man on bench`, `clear glasses on woman`, etc. The authors of [7] selected these objects and attributes from the most frequently occurring objects and attributes in the training set. For any given partial query, there is generally more than one matching image, rather than the single definitive match that exists in the full scene graph task. Because of this, the R@k and median rank calculations must be adjusted to account for multiple images. For R@k, I use the modified methodology from [7]. For a given k, $R = 1$ when *any* image from the positive set is retrieved, and 0 otherwise. As before, recall results for each query are the fraction of correct image retrievals over the top k results for the n queries.

The three methods produce identical results for the partial scene graph queries, which is to be expected, because all partial scene graph queries are constructed from the most frequently occurring objects. The unrecognized scoring models are only used when there are objects that are not detected by the R-CNN, thus they should function identically. The performance on partial scene graphs does reinforce that the unrecognized scoring models are only used when encountering unrecognized objects, and also sets a baseline performance for the next section where R@k performance of partial scene graphs with unrecognized objects is investigated. The results in Figure 26 and Table 3 are very similar to the values reported in [7]. The differences are likely related to the fact that I did not have access to the exact same query set as used in [7], which describes a data set of 598 partial scene graphs, with 119 randomly selected for testing. The set of graphs from the supplemental website contains 150 partial scene graphs. Additionally, my set of positive images may not be exactly the

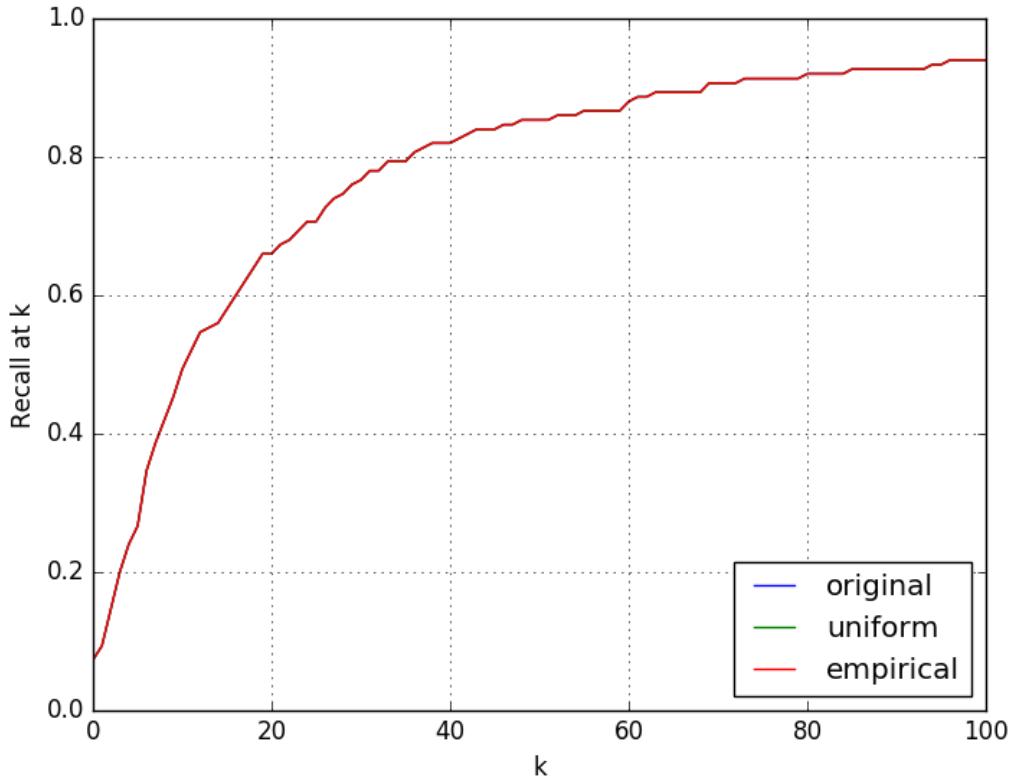


FIGURE 26: Recall at k for Partial scene graphs. The values are identical for all three methods, because the partial scene graph dataset is built from recognized objects. If only recognized objects are in the scene graph query, the unrecognized scoring methods will not be used, so the results should be identical for all three methods.

same, I followed the methodology described in [7] for determining the positive set of images, but cannot be completely certain that the sets are identical.

Although these results are not identical to those reported in [7], they are very similar to the plotted R@k data and the median rank value is identical.

	Original	Empirical	Uniform
Median r	11	11	11
Average r	29.413	29.413	29.413
Std Dev r	55.032	55.032	55.032
R@1	0.073	0.073	0.073
R@5	0.240	0.240	0.240
R@10	0.453	0.453	0.453
R@55	0.860	0.860	0.860
R@60	0.867	0.867	0.867
R@65	0.893	0.893	0.893

TABLE 3: Median rank and R@k values for partial scene graph retrieval. The partial scene graph data is constructed from recognized objects, attribute, and relationships, so performance of the three methods is identical

4.3 OBSCURED SCENE GRAPH RETRIEVAL

In order to investigate the performance of the unrecognized scoring models in a partial scene graph retrieval task, I modified the partial scene graph dataset described in Section 4.2 to ensure that an unrecognized object is present in the scene graphs; see Figure 27 for an example. I refer to this modified scene graph as an *obscured* scene

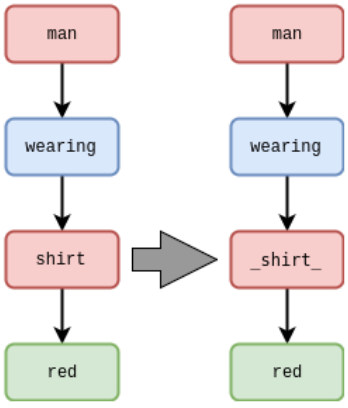


FIGURE 27: Obscuring a partial scene graph. The recognized object `shirt` is altered to an unrecognized class `_shirt_`. The partial scene graph dataset is constructed of only recognized object classes, so this obscured set allows testing R@k for the unrecognized scoring models on partial scene graphs.

graph. I alter the partial scene graph dataset by selecting one of the two objects in

each of the 150 scene graphs, and add an underscore character to the front and end of the class name e.g. `glasses` is changed to `_glasses_`, `car` is changed to `_car_`. Each query in the partial scene graph dataset is composed of two objects, one object always has an attribute (`red shirt`, `white keyboard`, etc.) and the other object will not. Given the choice of obscuring the object with no attribute or the object with an attribute, I obscure the class name of the object with an attribute, as shown in Figure 27. The object with an attribute is selected because this will produce the most difference in performance between the original method and the unrecognized object scoring methods. The original method discards the unrecognized object, and as such, it does not model the attribute or the relationship in the partial scene graph. The unrecognized object scoring methods will provide scores for the unrecognized object so that the factor graph can use the attribute and relationship scores when determining similarity.

The R@k plot and table results from running the obscured scene graphs are shown in Figure 28 and Table 4. For the obscured scene graph image retrieval task, performance of the three methods shows more variation than was seen in the full scene graph task. Each of the three methods has the best performance in some segment of the low k region: the empirical method is best at $k = 1$, the uniform method is best at $k = 5$, and the original method is best at $k = 10$. As in the full scene graph task, the empirical method shows the best recall in the $50 \leq k \leq 70$ regime. This is an interesting similarity, and would be particularly noteworthy if further research could identify what causes this improved performance and whether it can be extended to improve performance over the lower k ranks.

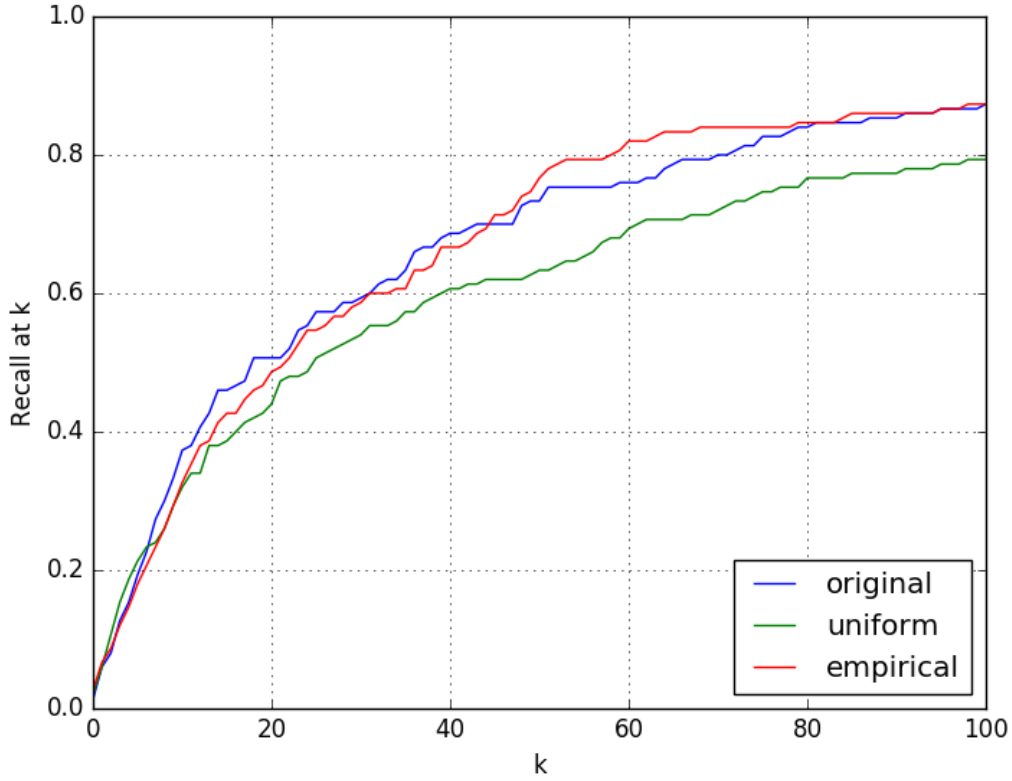


FIGURE 28: Recall at k for the obscured scene graphs. Best performance is mixed at the low k values, and the empirical scoring model is best in the mid-range k values, as seen for the full scene graph task.

	Original	Empirical	Uniform
Median r	**18**	22	25
Average r	50.940	**45.560**	72.060
Std Dev r	83.089	**71.318**	109.412
R@1	0.013	**0.027**	0.020
R@5	0.153	0.147	**0.187**
R@10	**0.333**	0.293	0.293
R@55	0.753	**0.793**	0.647
R@60	0.760	**0.807**	0.680
R@65	0.780	**0.833**	0.707

TABLE 4: Median rank and R@k values for the obscured scene graph image retrieval task. The best performing method varies in the low k values, and the empirical method shows best performance in the $50 \leq k \leq 70$ range.

Chapter 5

conclusions

The IRSG method for image retrieval generates solid results: a median rank of 22 or less for all methods, indicating that the correct images are likely to be found within the 30 images that one would see on the first page of a set of image search results. The authors of [7] envisioned the IRSG as a framework for image retrieval, with a pipeline composed of object identification leading to object/attribute scoring and then to spatial relationship scoring. The image scoring functions are used to quantify a similarity measure with the scene graph query through inference on a graphical model. This similarity measure serves as a ranking for image retrieval.

In this thesis, I attempt to improve retrieval performance through a scoring model that can provide scores for unrecognized objects in a scene graph query. I tested the new scoring models in the IRSG framework to determine if they improve recall performance. The new models did not help to improve performance in the most important area of performance: the first 30 images. Despite not improving performance for low k values, the empirical scoring model does show promise in the $50 \leq k \leq 70$ ranks, which is an interesting avenue for further research. The uniform method performed most poorly overall, likely because the object scoring values were so spread out across the candidate image bounding boxes that inference was not able to compensate for

the uninformative distribution by the extra information gained in the attribute and relationship factors. The empirical scoring model shows better performance than the uniform scoring model, likely because the empirical scoring method's averaging of recognized objects is marginally more informative than the scores generated by the uniform scoring model. It may yet prove useful to investigate other methods of accounting for attributes and relationships in the framework that allow the model to use the attribute and relationship information without reducing the overall predictive power of the original method.

Chapter 6

future work

This image retrieval framework presents an interesting technique for combining a set of models to retrieve images by attempting to leverage the semantic content of the images. The image's semantic information is determined by modeling not just the objects but the attributes and relationships within the image. There is ample opportunity for further study of this method to improve retrieval performance and generalize the existing configuration.

Of primary interest is to investigate what factors contribute to the framework generating good results: low energy for well-matching images, high energy for poorly-matching images. Naturally, this line of investigation should include an examination of the factors that contribute to bad results: high energy for well-matching images and low energy for poorly-matching images.

The method is very reliant on good quality bounding boxes. It would be of interest to investigate more recent techniques in image segmentation such as Fully Convolutional Networks for Semantic Segmentation by Long et al. [14]. Improved object boxes could potentially provide a better starting point for the factor graph when running inference to generate energy values. It is even possible that the object proposal and

classification components can be combined into one model using recent techniques [**?**], and it would be interesting to see how performance is affected using these combined segmentation and classification models.

Currently, IRSG weights the unary and binary components equally, but a learned or configured weighting parameter could be added to the model to control the factors' contribution to the overall value such as λ in the equation below:

$$P(\gamma \,|\, G, B) = \lambda \prod_{o \in O} P(o \,|\, \gamma_o) \; (1 - \lambda) \prod_{(o,r,o') \in E} P(\gamma_o, \gamma_{o'} \,|\, o, r, o')$$

Would allowing the unary factors to have more influence over the result provide improved performance? What if the binary factors have more weight? Would any performance increase be consistent across the full range of k or would it be localized? There are many questions raised by this alteration, and it would be a very interesting avenue for further research.

Another question is whether the semantics learned by the framework from the MS COCO & YFCC100m are transferable to other image sets. It is presumed that the method is learning generalized features of attribute, object, and relationship detection, and that it will transfer to other data. It would be useful to ensure that the method can perform reasonably on datasets outside of the 5,000 semantic configurations that it had been trained upon. The image retrieval framework presents an interesting technique for deriving semantic meaning from images, and then using that learned semantic understanding to infer how well a scene graph query matches an image or set of images. This framework provides a general pipeline that provides several opportunities for further improvement and enhancement.

References

[1] Andres, B., Beier, T., and Kappes, J. H. Opengm: A c++ library for discrete graphical models, 2012.

[2] Carlson, A., Betteridge, J., Kisiel, B., Settles, B., Hruschka, E. R., and Mitchell, T. M. Toward an architecture for never-ending language learning. In *In AAAI* (2010).

[3] Chang, A. X., Savva, M., and Manning, C. D. Learning spatial knowledge for text to 3d scene generation. In *EMNLP* (2014), pp. 2028–2038.

[4] Chen, X., Shrivastava, A., and Gupta, A. Neil: Extracting visual knowledge from web data. In *Proceedings of the IEEE International Conference on Computer Vision* (2013), pp. 1409–1416.

[5] Girshick, R. B., Donahue, J., Darrell, T., and Malik, J. Rich feature hierarchies for accurate object detection and semantic segmentation. *CoRR abs/1311.2524* (2013).

[6] Hodosh, M., Young, P., and Hockenmaier, J. Framing image description as a ranking task: Data, models and evaluation metrics. *J. Artif. Int. Res. 47*, 1 (May 2013), 853–899.

[7] Johnson, J., Krishna, R., Stark, M., Li, L.-J., Shamma, D., Bernstein, M., and Fei-Fei, L. Image retrieval using scene graphs. In *Proceedings*

of the IEEE Conference on Computer Vision and Pattern Recognition (2015), pp. 3668–3678.

[8] Kappes, J. H., Andres, B., Hamprecht, F. A., Schnörr, C., Nowozin, S., Batra, D., Kim, S., Kausler, B. X., Kröger, T., Lellmann, J., Komodakis, N., Savchynskyy, B., and Rother, C. A comparative study of modern inference techniques for structured discrete energy minimization problems. *CoRR abs/1404.0533* (2014).

[9] Karpathy, A., and Fei-Fei, L. Deep visual-semantic alignments for generating image descriptions. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (June 2015).

[10] Koller, D., and Friedman, N. *Probabilistic Graphical Models: Principles and Techniques - Adaptive Computation and Machine Learning*. The MIT Press, 2009.

[11] Krähenbühl, P., and Koltun, V. Geodesic object proposals. In *European Conference on Computer Vision* (2014), Springer, pp. 725–739.

[12] Kschischang, F. R., Frey, B. J., and Loeliger, H.-A. Factor graphs and the sum-product algorithm. *IEEE Transactions on information theory 47*, 2 (2001), 498–519.

[13] Lin, T., Maire, M., Belongie, S. J., Bourdev, L. D., Girshick, R. B., Hays, J., Perona, P., Ramanan, D., Dollár, P., and Zitnick, C. L. Microsoft COCO: common objects in context. *CoRR abs/1405.0312* (2014).

[14] Long, J., Shelhamer, E., and Darrell, T. Fully convolutional networks for semantic segmentation. *CoRR abs/1411.4038* (2014).

[15] Mitchell, M., Han, X., Dodge, J., Mensch, A., Goyal, A., Berg, A., Yamaguchi, K., Berg, T., Stratos, K., and Daumé, III, H. Midge: Generating image descriptions from computer vision detections. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics* (Stroudsburg, PA, USA, 2012), EACL '12, Association for Computational Linguistics, pp. 747–756.

[16] Murphy, K. P. *Machine Learning: A Probabilistic Perspective*. The MIT Press, 2012.

[17] Platt, J. C. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In *ADVANCES IN LARGE MARGIN CLASSIFIERS* (1999), MIT Press, pp. 61–74.

[18] Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C., and Fei-Fei, L. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV) 115*, 3 (2015), 211–252.

[19] Thomee, B., Shamma, D. A., Friedland, G., Elizalde, B., Ni, K., Poland, D., Borth, D., and Li, L.-J. YFCC100M: The new data in multimedia research. *Communications of the ACM 59*, 2 (2016), 64–73.

[20] Vinyals, O., Toshev, A., Bengio, S., and Erhan, D. Show and tell: A neural image caption generator. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015* (2015), pp. 3156–3164.

[21] Zitnick, C. L., and Parikh, D. Bringing semantics into focus using visual abstraction. In *Proceedings of the 2013 IEEE Conference on Computer Vision and*

Pattern Recognition (Washington, DC, USA, 2013), CVPR '13, IEEE Computer Society, pp. 3009–3016.