Image Similarity using Scene Graphs

A B. Tech Project Report Submitted
in Partial Fulfillment of the Requirements
for the Degree of

Bachelor of Technology

by

Anubhav Tyagi
(170101009)

under the guidance of

Dr. Amit Awekar

to the

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY GUWAHATI
GUWAHATI - 781039, ASSAM

CERTIFICATE

*This is to certify that the work contained in this thesis entitled "**Image Similarity using Scene Graphs**" is a bonafide work of **Anubhav Tyagi** (**Roll No. 170101009**), carried out in the Department of Computer Science and Engineering, Indian Institute of Technology Guwahati under my supervision and that it has not been submitted elsewhere for a degree.*

Supervisor: **Dr. Amit Awekar**

Assistant Professor,

April, 2021

Department of Computer Science & Engineering,

Guwahati.

Indian Institute of Technology Guwahati, Assam.

Acknowledgements

I would like to thank my supervisor, Dr. Amit Awekar for his support during the course of the Bachelors Thesis Project.

Contents

List of Figures **vii**

List of Tables **ix**

1 Introduction **1**

1.1 Abstract 1

1.2 Introduction 2

1.3 Organization of The Report 2

2 Review of Prior Works **3**

2.1 Efficient Graph Similarity Joins with Edit Distance Constraints 3

2.2 Vertex-Edge Overlap (VEO) 4

2.3 Conclusion 5

3 Dataset & Preprocessing **7**

3.1 Dataset Description 7

3.1.1 Mapillary Vistas Dataset 7

3.1.2 Visual Genome Dataset 8

3.2 Scene Graph Construction 10

3.2.1 Mapillary Vistas Dataset 11

3.2.2 Visual Genome Dataset 11

3.3 Conclusion 12

4 Algorithms **13**

4.1 Graph similarity join with edit distance constraints 13

4.1.1 Results 14

4.2 Vertex Edge Overlap 14

4.2.1 Results 15

4.3 Vertex Edge Overlap with partial matching 16

4.3.1 Additional Improvements 18

4.3.2 Results 18

4.4 Conclusion 18

5 Limitations & Future Work **21**

5.1 Limitations 21

5.2 Future Work 21

References **23**

List of Figures

3.1 Example from Mapillary Vistas Dataset 8

4.1 Similar image pairs for various τ values 14

4.2 VEO result on mapillary vistas dataset 15

4.3 VEO result on visual genome dataset 16

4.4 Top 5 results for VEO with partial matching on visual genome dataset . . 19

5.1 Incorrect image annotation in visual genome dataset 22

List of Tables

3.1 Mapillary Vistas Dataset description 8
3.2 Visual Genome Dataset description 9
3.3 Average statistics in Visual Genome Dataset 9
3.4 Commonly occurring classes in Visual Genome Dataset 10

Chapter 1

Introduction

1.1 Abstract

Image similarity has been actively studied for many years. Most state-of-art approaches for image similarity use deep neural networks to address the problem. However, deep neural net frameworks require huge training time. In this paper, a method is described to represent an annotated image as a scene graph, that captures the semantics of the image. The scene graph is a representation of the image as an undirected graph, where each vertex in the graph corresponds to a object in the image and the edge between two vertices describes the relationship between the corresponding objects. In this paper, an algorithm is proposed that computes the extend of similarity between the images. The algorithm takes as input two images, constructs the scene graph for each image as an intermediary step and finally outputs the similarity score.

1.2 Introduction

The notion of similarity between images is defined in terms of nearness in the feature set describing each image. The problem of image similarity has plenty of practical applications such as social media analysis, web-image search, digital forensics and online shopping to name a few. Classical web image retrieval requires searching for visually similar images depicting the same content as the query image. With the growth in popularity and amount of multimedia content on social media, businesses can analyse the posts of users and accordingly plan marketing strategies.

We adopt a very broad definition of similar images, encompassing all images of the same object or scene. The standard hashing based techniques cannot be used for finding similar images, since even minimal alterations, such as cropping and resizing, would make different copies of the same image untraceable.

Image similarity techniques are desired to retrieve semantically similar images within a database based on a query image. The task of image similarity can be divided into two sub-parts : describing image content in terms of visual features and studying efficient similarity functions over the features. The features from images can be learned by using deep Convolution Neural Networks.

1.3 Organization of The Report

This chapter provides a background of the topics covered in this report. The introduction explains the problem statement and its applications in various domains. We review prior works in the domain of graph similarity and various similarity functions that have been proposed.In chapter 3, we provide a analysis of the datasets used for our experiments and also describe the method for converting annotated images to scene graphs. In chapter 4, we discuss the implementation details of the algorithms and discuss the results obtained on those algorithms. And finally in chapter 5, we conclude with limitations and future work.

Chapter 2

Review of Prior Works

The chapter deals with the review of some of the graph similarity metrics and corresponding algorithms described in prior papers.

2.1 Efficient Graph Similarity Joins with Edit Distance Constraints

The subsection provides an overview of [ZXLW12]

The graph edit distance between r and s is the minimum number of edit operations that transform r to a graph isomorphic to s.The edit operations consist of insertions, deletions and renaming labels of vertices/edges. The computation of graph edit distance between two graphs is NP-hard. In order to tackle NP-hardness of the problem, the paper proposes a graph similarity join problem with edit distance constraints.

Inspired by the idea of q-gram on string similarity, the paper proposes $GSimJoin$ algorithm that defines q-gram on graphs based on path. A path based q-gram in a graph is a simple path of length q. The idea of the edit distance constraints is that an edit operation will only affect a limited number of q-grams.The maximum number of q-grams that can be

affected by an edit operation is shown as

$$D_{path} = max_{u \epsilon V(r)} |Q_r^u|$$

where Q_r^u denotes the multiset of q-grams that contain vertex u and $V(r)$ denotes the vertex set in graph r. It then defines a count filtering condition for the path-based q-grams that two graphs,r and s must share atleast

$$LB_{path} = max(|Q_r| - \tau D_{path}(r), |Q_s| - \tau D_{path}(s))$$

common q-grams if they are within graph edit distance τ. A pair of graph that satisfies the lower bound condition form a candidate pair. The algorithm uses prefix filtering to find candidate pairs that satisfy count filtering condition. According to path filtering, let Q_r and Q_s are q-gram multisets for r and s respectively sorted according to global ordering of the q-gram universe. If $|Q_r \cap Q_s| \geq \alpha$, then the $(Q_r - \alpha + 1)$-prefix of Q_r and the $(Q_s - \alpha + 1)$-prefix of Q_s, must have atleast one common q-gram. Rare q-grams are favoured in prefixes in order to achieve small candidate size and fast execution. The algorithm then uses minimum edit filtering and label filtering using mismatching q-gram to further reduce the candidate size. It then invokes expensive graph edit calculation for every candidate pair that survive all the filtering conditions to tell if they are join results.

2.2 Vertex-Edge Overlap (VEO)

The subsection provides an overview of Vertex/edge overlap from the work [PPGM10]

The idea of similarity for Vertex Edge Overlap is based on the rule that "two graphs are similar if they share many vertices and edges". Graph edit distance and Jaccard Index are two main ways of computing this kind of similarity. We discussed graph edit distance in the previous section. Jaccard index is defined as the intersection of vertices/edges divided

by the union of vertices/edges. The similarity function for two graphs G and G' is defined as

$$sim_{VEO}(G, G') = 2\frac{|V \cap V'| + |E \cap E'|}{|V| + |V'| + |E| + |E'|}$$

The time complexity for algorithm implementing sim_{VEO} is $O(|V| + |V'| + |E| + |E'|)$.The algorithm implementing this scheme is called Vertex/Edge Overlap algorithm.

2.3 Conclusion

This chapter provides details of some of the existing graph similarity metrics and algorithms. In the next chapter, we provide an analysis of the datasets used in our experiments and methods to convert annotated images from those datasets to scene graphs, that will be used as input to the above algorithms.

Chapter 3

Dataset & Preprocessing

In this chapter, we provide a description of the datasets used. We also describe the preprocessing of datasets to construct scene graphs which will be used as input for our algorithm. We then describe the improvements to base algorithms from next chapter onwards.

3.1 Dataset Description

In this section, we describe the datasets used for running the experiments. We use two annotated image datasets namely Mapillary Vistas Dataset and Visual Gnome Dataset. We provide a brief description of each one of these in the following subsections.

3.1.1 Mapillary Vistas Dataset

In this section, we provide a brief description of Mapillary Vistas Dataset [NORBK17] and describe the process of using the dataset. The dataset can be downloaded from `https://www.mapillary.com/dataset/vistas`. The dataset contains images segmented into training, validation and testing subsets. For our purpose, we use the images in the training subset since this subset contains annotated images. The training subset contains a total of 18000 images with 66 semantic object categories. A brief summary of the dataset is given in Table 3.1

Number of images	18000
Number of object categories	66

Table 3.1: Mapillary Vistas Dataset description

Each semantic object class is represented by a specific colour in the image. The colour representing each object class remains constant throughout the dataset. The information regarding the semantic object class and the corresponding colour representing the object class can be found in config.json file, that is downloaded along with the mapillary vistas dataset. A sample annotated image from the dataset is shown in Figure 3.1

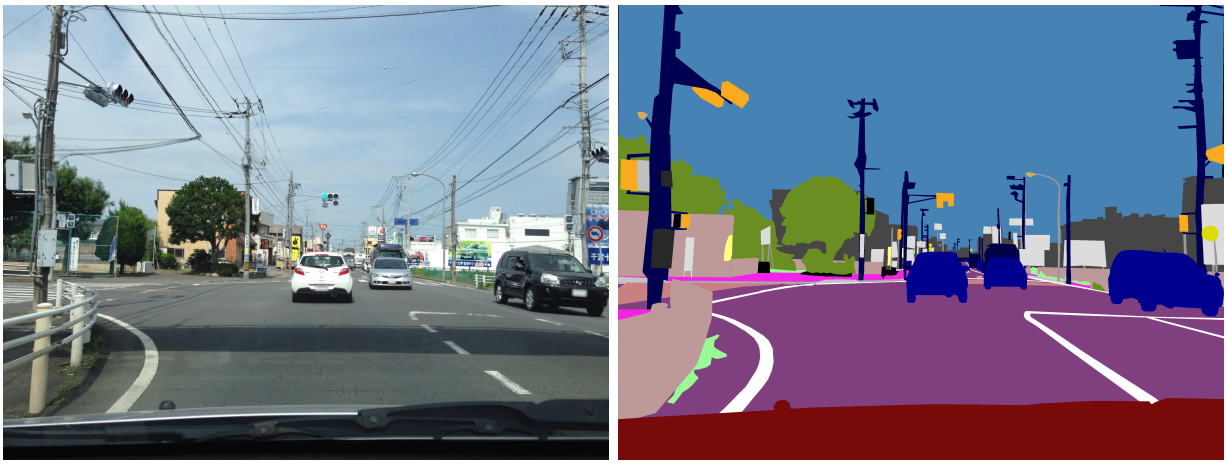


Fig. 3.1: Example from Mapillary Vistas Dataset

3.1.2 Visual Genome Dataset

In this section, we provide a brief description of Visual Genome Dataset [KZG+16] and describe the process of using the dataset. The dataset can be downloaded from `https://visualgenome.org/api/v0/api_home.html`. For our experiments, we use v1.2 of the dataset. The dataset contains 108,077 images. The dataset in addition to the images, contains *objects.json* and *relationships.json*. objects.json is structured as a list of json

objects, where each element of the list stores the image id and the list of objects associated with the particular image recognised by the image id. relationships.json is also structured as a list of json objects, where each element of the list stores the image id and a list containing all the relationships recognised between the objects in the image. A brief summary of the dataset is given in Table 3.2

Number of images	108077
Number of distinct object classes	5996
Number of distinct relationship classes	1014

Table 3.2: Visual Genome Dataset description

The average distribution statistics of objects and relationships across images is summarized in Table 3.3.

Average number of objects per image	32.34
Average number of relationships per image	16.81

Table 3.3: Average statistics in Visual Genome Dataset

The objects and relationship labels are assigned to images manually. The labels, on one side being more reliable since human intelligence is involved instead of machine learning frameworks, are also prone to human errors. These errors may result in incorrect results in few cases which we will discuss along with the results. The most commonly occurring objects and relationship classes are listed in Table 3.4.

Most common object classes	Most common Relationship classes
man	ON
person	has
woman	IN
building	wears
sign	behind
table	next to

Table 3.4: Commonly occurring classes in Visual Genome Dataset

3.2 Scene Graph Construction

In this section, we describe our method to construct scene graphs for the above mentioned datasets. Since the organization and structure of both the datasets are different, we employ different methods to construct scene graphs for each dataset. Scene graph is a simple undirected graph that captures the semantics of the image scene. Each node in the scene graph, represents a specific object in the image. The nodes in the graph are labelled based on the object class it belongs to. An edge between two nodes in a graph points to a relation between the two objects in the image. The edges may be labelled or unlabelled depending on the quality of information that the dataset provides.The following subsections provide an elaborate description of the scene graph construction methods for both the datasets.

3.2.1 Mapillary Vistas Dataset

Each image in the dataset can be visualized as a two-dimensional array of pixels, where each pixel stores the RGB colour value at that particular point in image. We run Breadth First Search (BFS) to find continuous regions. A continuous region is a collection of pixels such that for any two pixels in the region, there exists a path from one pixel to another and all pixels on the path have same colour as the two end point pixel cells. According to the definition of node in scene graph, each continuous region in a image forms a single node in the graphs. Since defining nodes using continuous regions will introduce a lot of noise, we only choose continuous regions whose area is atleast 0.01% of the total image area. Thus the total number of nodes in the scene graph is equal to the number of continuous regions in the annotated image. Two nodes in the scene graph share an edge if and only if the corresponding continuous regions share a pixel boundary. Since there is no specific relationship that can be inferred between the two nodes, the edges are unlabelled. However, since unlabelled edges do not impart any useful information in graph similarity metrics, we provide a label to an edge using a function of labels of the corresponding nodes that it connects. Thus, for n number of distinct object classes in the dataset, there are a total of $\binom{n}{2}$ distinct edge labels possible.

3.2.2 Visual Genome Dataset

The dataset provides objects and relationship labels as json files. Thus constructing scene graph for visual genome dataset requires parsing the json files. The objects.json file contains information required for constructing the nodes in the scene graph and characterizing them to one of the pre-defined object class. The relationships.json file stores information required for adding edges to the scene graph. Each relation in the relationship.json has an associated predicate with it, that provides basis for assigning the edges in the graph to one of the pre-defined relationship class. The information from both the files is merged to construct a scene graph for an image. One thing to note here is that the labels for objects and relations

are of data type strings. Thus before constructing the scene graphs, we assign a numeric label to each object and relation class.

3.3 Conclusion

In this chapter, we provide a brief analysis of mapillary vistas dataset and visual genome dataset used for our experiments. We also describe the method for conversion of annotated images in both the datasets to scene graphs. For each dataset, we create a single file that stores the scene graphs for all the images in the dataset. This single file, serves as a input to the algorithms that we discuss in the next chapter. In the next chapter, we discuss the graph similarity algorithms and their implementation in detail.

Chapter 4

Algorithms

In the previous chapter, we described the method for creation of scene graphs. The output of this is a single file containing scene graphs for all the images in a dataset. This file will serve as a input to our algorithm.

4.1 Graph similarity join with edit distance constraints

The following section describes in brief the graph similarity join algorithm with edit distance constraints. The idea behind the algorithm is described briefly in Section 2.1. The implementation of the code is borrowed from the paper [ZXLW12].

The algorithm takes as input a set of graphs, r, and maximum permissible edit distance, τ, and returns as output the list of graphs that satisfy the edit distance constraints. The algorithm first uses prefix filtering to implement count filtering condition to get possible candidate pairs. The algorithm then uses minimum edit filtering and local label filtering to estimate the lower bound of the graph edit distance for the candidate pairs based on mismatching q-grams. If the lower bound for a candidate pair is greater than than the maximum permissible graph edit distance τ, the candidate pair is discarded. The remaining candidate pairs are then subjected to expensive graph edit distance calculation to output the final pairs.

4.1.1 Results

We run the graph similarity join with edit distance constraints on mapillary vistas dataset. A few results for various graph edit distance threshold are shown in Figure 4.1

(a) $\tau = 1$

(b) $\tau = 3$

Fig. 4.1: Similar image pairs for various τ values

4.2 Vertex Edge Overlap

The following section explains the implementation of Vertex Edge Overlap algorithm. The pseudo code for the algorithm in Algorithm 1

The algorithm takes as input two graphs $G(V, E)$ and $G'(V', E')$. The vertex and edge list are sorted according to their label id's. Line 1 of the algorithm finds the intersection of vertex set of the two graphs. Since the vertex list is sorted based on the label id of the vertices, this step takes $O(|V| + |V'|)$ time steps using two pointer method. Line 2 of

Algorithm 1: Vertex Edge Overlap

Input: Graph G and G' with vertices and edges sorted by label Id. Let V and V' denote the vertex list and E and E' denote the edge list for G and G' respectively

Output: Similarity score $\in$ [0,1]

1 CV $= V \cap V'$;

2 CE $= E \cap E'$;

3 $similarityScore = 2\times \frac{|CV|+|CE|}{|V|+|V'|+|E|+|E'|}$

the algorithm finds the intersection of edge set of the two graphs. Since the edge list is sorted based on label id of edges, this step takes $O(|E|+|E'|)$ time steps using two pointer method. Finally the similarity score is calculated using the formula as explained in Section 2.2. The overall time complexity for the algorithm is $O(|V|+|V'|+|E|+|E'|)$.

4.2.1 Results

We run the vertex edge overlap algorithm on mapillary vistas dataset and visual genome dataset. A few results on mapillary vistas dataset are shown in Figure 4.2

(a) Similarity Score = 0.86

(b) Similarity Score = 0.83

Fig. 4.2: VEO result on mapillary vistas dataset

A few results of vertex edge overlap algorithm on visual genome dataset are shown in Figure 4.3

(a) Similarity Score = 1.0

(b) Similarity Score = 0.91

Fig. 4.3: VEO result on visual genome dataset

4.3 Vertex Edge Overlap with partial matching

In vertex edge overlap algorithm, only exact matching vertices and edges contribute in the final similarity score. However, when comparing scene graphs of two images, partial match score of vertices must also be incorporated while calculating the final similarity score. This can be understood by the fact that two different object classes may represent semantically similar objects. For instance, person and man, and pants and trousers represents semantically similar objects. We now elaborate the complete algorithm along with our modification. The algorithm is summarized in Algorithm 2

The algorithm takes as input two graphs $G(V, E)$ and $G'(V', E')$. The vertex and edge list are sorted according to their label id's. Line 1-2 of the algorithm finds the intersection

Algorithm 2: Vertex Edge Overlap with partial matching

Input: Graph G and G' with vertices and edges sorted by label Id. Let V and V' denote the vertex list and E and E' denote the edge list for G and G' respectively

Output: Similarity score $\in$ [0,1]

1 CV $= V \cap V'$;
2 CE $= E \cap E'$;
3 NCV_$G = V -$ CV ;
4 NCV_$G' = V' -$ CV ;
5 Construct 2D-array $costMatrix$ of size |NCV_G| $\times$ |NCV_G'| where $costMatrix[i][j]$ denotes the partial similarity score between vertex labels NCV_$G[i]$ and NCV_$G'[j]$
6 $maxWeightedMatching = 0$
7 **for** $i = 1$ *to* P **do**
8 Randomly shuffle the rows of $costMatrix$
9 $matchingScore$ = Starting from $1st$ row, greedily choose the maximum weighted matching score ;
10 $maxWeightedMatching = max($ $maxWeightedMatching$, $matchingScore$ $)$
11 **end**
12 $similarityScore = 2\times \frac{|CV|+|CE|+maxWeightedMatching}{|V|+|V'|+|E|+|E'|}$;

vertex and edge list and is implemented using the tow pointer method. Line 3-4 finds the mismatched vertex and edge labels in the two graphs. This is again implemented using two pointer method and thus takes time linear in the number of vertices and edges.

Line 5-11 implements the partial matching of vertices. Line 5 construct a 2D-array $costmatrix$ of size |NCV_G| $\times$ |NCV_G'|. The element at indexed at (i, j) in $costMatrix$, is the partial similarity score between vertex label NCV_$G[i]$ and NCV_$G'[j]$. To calculate the partial similarity score between vertex labels, which are originally strings as described in Section 3.2.2, we use binary word embedding from fasttext [MGB$^+$18]. Using fasttext, each label is then transformed into a 300-dimensional vector. The cosine similarity between the vectors of the corresponding labels are used as partial similarity score between the vertex labels. The algorithm then uses a greedy approach to calculate the maximum weighted matching. Line 6-11 implements this part of the algorithm. At each iteration, we randomly shuffle the rows of the $costMatrix$, and then starting from first row, greedily calculate the

maximum possible weighted matching in the given permutation.

The similarity score is then calculated using the similar formula as in simple Vertex Edge Overlap, except the fact that partial similarity score between vertices is also taken into account.

4.3.1 Additional Improvements

The greedy maximum weighted matching approach returns values close to the maximum weighted matching score in most cases but it does not return the optimal maximum weighted matching score in some of the cases. The problem of maximum weighted matching is implemented by the Hungarian Algorithm [Kuh55]. We tried using Hungarian algorithm in place of greedy approach, but since the time complexity of Hungarian Algorithm is $O(V^3)$, the VEO with partial matching takes around 300 days to obtain results for pair-wise similarity score for all possible image pairs in visual genome dataset. Thus, we only stick to greedy approach that yields results close to the optimal value and has a time complexity of $O(V^2)$.

4.3.2 Results

We run vertex edge overlap with partial matching on visual genome dataset. Top 5 image pairs with highest similarity scores are shown in Figure 4.4

4.4 Conclusion

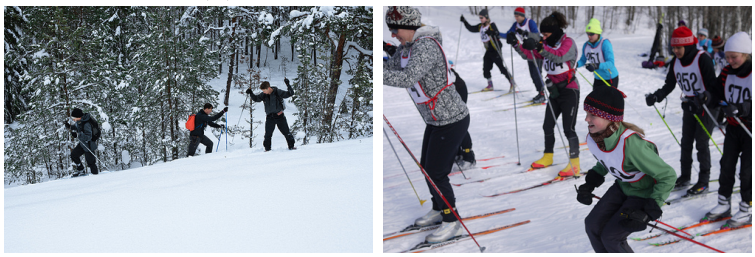
In this chapter, we discuss the implementation details of various algorithms used and results obtained by running these algorithms on the datasets. In the next chapter, we discuss the limitations of the datasets and algorithms and discuss the future work that can pave way for a finer similarity function.

(a) Similarity Score = 1.0

(b) Similarity Score = 1.0

(c) Similarity Score = 0.933

(d) Similarity Score = 0.931



(e) Similarity Score = 0.917

Fig. 4.4: Top 5 results for VEO with partial matching on visual genome dataset

Chapter 5

Limitations & Future Work

5.1 Limitations

There are few limitations of the dataset that limit the quality of results that are obtained. The mapillary vistas dataset, does not contain relationship annotations between objects which leads to semantically distinct image pairs being labelled as similar by the algorithm. The visual genome dataset, which has human annotated images, has incorrectly labelled images due to human error. Since the algorithms uses these labels to characterize the images as similar, this results in quite distinct images being labelled as similar. One such example of incorrect annotation of images that leads to high similarity score is shown in Figure 5.1. In the second sub figure, the aeroplane is incorrectly labelled as train, which results in high similarity score between the image pairs.

A better quality dataset is thus required to improve and assess the quality of results obtained by the algorithms described in previous chapter.

5.2 Future Work

The limitations with datasets can be resolved by captioning images using deep neural networks like $DenseCap: Fully\,Convolutional\,Localization\,Networks\,for\,Dense\,Captioning$

(a) Similarity Score = 0.92

Fig. 5.1: Incorrect image annotation in visual genome dataset

[JKL15]. Natural language processing can be applied to these machine generated captions to provide suitable labels to the image objects and their corresponding relationships.

Moreover, simple Vertex Edge Overlap and Vertex Edge Overlap with partial matching imparts equal weight to every label type during calculation of similarity score. However, certain label classes occur more frequently across images than others and thus must be given smaller weight in computation of similarity score.

References

[JKL15] Justin Johnson, Andrej Karpathy, and Fei-Fei Li. Densecap: Fully convolutional localization networks for dense captioning. *CoRR*, abs/1511.07571, 2015.

[JKS+15] J. Johnson, R. Krishna, M. Stark, L. Li, D. A. Shamma, M. S. Bernstein, and L. Fei-Fei. Image retrieval using scene graphs. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3668–3678, 2015.

[Kuh55] H. W. Kuhn. The hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2(1-2):83–97, 1955.

[KZG+16] Ranjay Krishna, Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen, Yannis Kalantidis, Li-Jia Li, David A Shamma, Michael Bernstein, and Li Fei-Fei. Visual genome: Connecting language and vision using crowdsourced dense image annotations. 2016.

[MGB+18] Tomas Mikolov, Edouard Grave, Piotr Bojanowski, Christian Puhrsch, and Armand Joulin. Advances in pre-training distributed word representations. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018)*, 2018.

[ML19] Lia Morra and Fabrizio Lamberti. Benchmarking unsupervised near-duplicate image detection. *Expert Systems with Applications*, 135:313 – 326, 2019.

[NORBK17] Gerhard Neuhold, Tobias Ollmann, Samuel Rota Bulò, and Peter Kontschieder. The mapillary vistas dataset for semantic understanding of street scenes. In *International Conference on Computer Vision (ICCV)*, 2017.

[PPGM10] Ali Dasdan Panagiotis Papadimitriou and Hector Garcia-Molina. Web graph similarity for anomaly detection. *Journal of Internet Services and Applications*, 1:19 – 30, 2010.

[ST20] B. Schroeder and S. Tripathi. Structured query-based image retrieval using scene graphs. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 680–684, 2020.

[ZXLW12] X. Zhao, C. Xiao, X. Lin, and W. Wang. Efficient graph similarity joins with edit distance constraints. In *2012 IEEE 28th International Conference on Data Engineering*, pages 834–845, April 2012.

Turnitin Originality Report

Processed on: 17-Apr-2021 16:31 IST
ID: 1558896342
Word Count: 4575
Submitted: 2

BTP_Report By Anubhav TYAGI

Similarity Index	Similarity by Source	
19%	Internet Sources:	12%
	Publications:	14%
	Student Papers:	N/A

3% match (publications)
Xiang Zhao, Chuan Xiao, Xuemin Lin, Wei Wang. "Efficient Graph Similarity Joins with Edit Distance Constraints", 2012 IEEE 28th International Conference on Data Engineering, 2012

3% match (Internet from 29-Jul-2014)
http://152.3.140.5/~abhinath/btp.pdf

1% match (publications)
Xiang Zhao, Chuan Xiao, Xuemin Lin, Wei Wang, Yoshiharu Ishikawa. "Efficient processing of graph similarity queries with edit distance constraints", The VLDB Journal, 2013

1% match (publications)
Aritra Ghosh, Pallavi Gudipati. "Anomaly detection in web graphs using vertex neighbourhood based signature similarity methods", 2016 International Conference on Data Science and Engineering (ICDSE), 2016

1% match (Internet from 07-Jul-2020)
https://www.hindawi.com/journals/tswj/2014/749028/

1% match (publications)
Lia Morra, Fabrizio Lamberti. "Benchmarking unsupervised near-duplicate image detection", Expert Systems with Applications, 2019

1% match (Internet from 25-Nov-2016)
https://pdfs.semanticscholar.org/57d0/15cb43b43913b54be7b5c73765b6382d7e99.pdf

1% match (publications)
Panagiotis Papadimitriou. "Web graph similarity for anomaly detection", Journal of Internet Services and Applications, 02/25/2010

1% match (Internet from 20-Jan-2010)
http://plg.uwaterloo.ca/~migod/supervision/CoryKapser-PhDThesis-2009.pdf

1% match (publications)
Guo Li, Guan Rong, Li Kenli, Li Renfa. "Fast Parallel Molecular Algorithms for DNA-Based Computation: Graph Isomorphism Problem", 2009 2nd International Conference on Biomedical Engineering and Informatics, 2009

1% match (Internet from 15-Dec-2019)
https://arxiv.org/pdf/1904.12218.pdf

< 1% match (Internet from 03-Sep-2020)
https://repository.tudelft.nl/islandora/object/uuid:d59e3379-7598-4064-8ffb-0db0c1726c35/datastream/OBJ/download

< 1% match (publications)
"Natural Language Processing and Chinese Computing", Springer Science and Business Media LLC, 2018

< 1% match (Internet from 23-Jul-2018)
http://eprints-phd.biblio.unitn.it/2888/1/PhD-Thesis.pdf

< 1% match (publications)
Yifan Chen, Xiang Zhao, Chuan Xiao, Weiming Zhang, Jiuyang Tang. "Efficient and Scalable Graph Similarity Joins in MapReduce", The Scientific World Journal, 2014

< 1% match (publications)
"ECAI 2020", IOS Press, 2020

< 1% match (publications)
Wang, Wei, Jianbin Qin, Xiao Chuan, Xuemin Lin, and Heng Tao Shen. "VChunkJoin: An Efficient Algorithm for Edit Similarity Joins", IEEE Transactions on Knowledge and Data Engineering, 2012.

< 1% match (Internet from 07-Feb-2019)
http://melodi.ee.washington.edu/people/bilmes/mypapers/all.txt

< 1% match (Internet from 27-Sep-2006)
http://wing.comp.nus.edu.sg/publications/theses/weiLuThesis.pdf

< 1% match (Internet from 01-Dec-2015)
http://journal.jaltcall.org/articles/9_2_Lu.pdf

< 1% match (publications)
"Computer Vision – ECCV 2018", Springer Science and Business Media LLC, 2018

< 1% match ()
https://eprints.mdx.ac.uk/13503/1/568743.pdf

< 1% match (publications)
Weiguo Zheng, Lei Zou, Xiang Lian, Jeffrey Xu Yu, Shaoxu Song, Dongyan Zhao. "How to Build Templates for RDF Question/Answering", Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data - SIGMOD '15, 2015

< 1% match (Internet from 10-Feb-2020)
https://link.springer.com/content/pdf/10.1007/978-3-319-49340-4.pdf

< 1% match ()
http://arxiv.org/abs/1502.07576

< 1% match (publications)
"Computer Vision – ECCV 2016", Springer Nature, 2016

< 1% match (publications)
Dawei Bu, Gail Tomlinson, Cheryl M. Lewis, Cindy Zhang, Eric Kildebeck, David M. Euhus. "An intronic polymorphism associated with increased XRCC1 expression, reduced apoptosis and familial breast cancer", Breast Cancer Research and Treatment, 2006

< 1% match (Internet from 09-Nov-2020)
https://gupea.ub.gu.se/bitstream/2077/66921/1/gupea_2077_66921_1.pdf

< 1% match (Internet from 31-Aug-2020)
https://www.ideals.illinois.edu/bitstream/handle/2142/108027/HUNG-THESIS-2020.pdf?isAllowed=y&sequence=1

< 1% match (Internet from 14-Nov-2020)
https://kobra.uni-kassel.de/bitstream/handle/123456789/2016110251246/DissertationAndreasWitsch.pdf?isAllowed=y&sequence=3

< 1% match ()

http://tuprints.ulb.tu-darmstadt.de/view/person/Kroemer=3AOliver_B=2E=3A=3A.html

< 1% match (Internet from 18-Dec-2019)
http://export.arxiv.org/pdf/1911.01138

< 1% match (Internet from 22-Nov-2009)
http://www.cs.umu.se/education/examina/Rapporter/TimoElverkemperMThesis.pdf

< 1% match (publications)
Yongjiang Liang, Peixiang Zhao. "Similarity Search in Graph Databases: A Multi-Layered Indexing Approach", 2017 IEEE 33rd International Conference on Data Engineering (ICDE), 2017

< 1% match (publications)
"Computer Vision – ECCV 2018", Springer Science and Business Media LLC, 2018

< 1% match (publications)
Jea, T.-Y.. "A minutia-based partial fingerprint recognition system", Pattern Recognition, 200510

< 1% match (publications)
Yuyu Guo, Jingkuan Song, Lianli Gao, Heng Tao Shen. "One-shot Scene Graph Generation", Proceedings of the 28th ACM International Conference on Multimedia, 2020

< 1% match (publications)
Rami Al-Rfou, Bryan Perozzi, Dustin Zelle. "DDGK: Learning Graph Representations for Deep Divergence Graph Kernels", The World Wide Web Conference on - WWW '19, 2019

Image Similarity using Scene Graphs A B. Tech Project Report Submitted in Partial Fulfillment of the Requirements for the Degree of Bachelor of Technology by Anubhav Tyagi (170101009) under the guidance of Dr. Amit Awekar to the DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING INDIAN INSTITUTE OF TECHNOLOGY GUWAHATI GUWAHATI - 781039, ASSAM 2 CERTIFICATE This is to certify that the work contained in this thesis entitled “Image Similarity using Scene Graphs” is a bonafide work of Anubhav Tyagi (Roll No. 170101009), carried out in the Department of Computer Science and Engineering, Indian Institute of Technology Guwahati under my supervision and that it has not been submitted elsewhere for a degree. April, 2021 Guwahati. Supervisor: Dr. Amit Awekar Assistant Professor, Department of Computer Science & Engineering, Indian Institute of Technology Guwahati, Assam. i ii Acknowledgements I would like to thank my supervisor, Dr. Amit Awekar for his support during the course of the Bachelors Thesis Project. iii iv Contents List of Figures List of Tables 1 Introduction 1.1 Abstract . . . . . 1.2 Introduction . . . . . 1.3 Organization of The Report . . . . . 2 Review of Prior Works 2.1 Efficient Graph Similarity Joins with Edit Distance Constraints . . . . . 2.2 Vertex-Edge Overlap (VEO) . . . . . 2.3 Conclusion . . . . . 3 Dataset & Preprocessing 3.1 Dataset Description . . . . . 3.1.1 Mapillary Vistas Dataset . . . . . 3.1.2 Visual Genome Dataset . . . . . 3.2 Scene Graph Construction . . . . . 3.2.1 Mapillary Vistas Dataset . . . . . 3.2.2 Visual Genome Dataset . . . . . 3.3 Conclusion..... v vii ix 1 1 2 2 3 3 4 5 7 7 7 8 10 11 11 12 4 Algorithms 4.1 Graph similarity join with edit distance constraints . . . . . 4.1.1 Results..... 4.2 Vertex Edge Overlap . . . . . 4.2.1 Results..... 4.3 Vertex Edge Overlap with partial matching . . . . . 4.3.1 Additional Improvements . . . . . 4.3.2 Results . . . . . 4.4 Conclusion..... 5 Limitations & Future Work 5.1 Limitations . . . . . 5.2 Future Work . . . . . References 13 13 14 14 15 16 18 18 18 21 21 21 23 vi List of Figures 3.1 4.1 4.2 4.3 Example from Mapillary Vistas Dataset . . . . . Similar

image pairs for various τ values . . . . . VEO result on mapillary vistas dataset . . . . . VEO result on visual genome dataset . . . . . 4.4 Top 5 results for VEO with partial matching on visual genome dataset . . 5.1 Incorrect image annotation in visual genome dataset . . . . . 8 14 15 16 19 22 vii viii List of Tables 3.1 Mapillary Vistas Dataset description . . . . . 3.2 Visual Genome Dataset description . . . . . 3.3 Average statistics in Visual Genome Dataset . . . . . 3.4 Commonly occurring classes in Visual Genome Dataset . . . . . 8 9 9 10 ix x Chapter 1 Intro duction 1.1 Abstract Image similarity has been actively studied for many years. Most state-of-art approaches for image similarity use deep neural networks to address the problem. However, deep neural net frameworks require huge training time. In this paper, a method is described to represent an annotated image as a scene graph, that captures the semantics of the image. The scene graph is a representation of the image as an undirected graph, where each vertex in the graph corresponds to a object in the image and the edge between two vertices describes the relationship between the corresponding objects. In this paper, an algorithm is proposed that computes the extend of similarity between the images. The algorithm takes as input two images, constructs the scene graph for each image as an intermediary step and finally outputs the similarity score. 1 1.2 Introduction The notion of similarity between images is defined in terms of nearness in the feature set describing each image. The problem of image similarity has plenty of practical applica- tions such as social media analysis, web-image search, digital forensics and online shopping to name a few. Classical web image retrieval requires searching for visually similar im- ages depicting the same content as the query image. With the growth in popularity and amount of multimedia content on social media, businesses can analyse the posts of users and accordingly plan marketing strategies. We adopt a very broad definition of similar images, encompassing all images of the same object or scene. The standard hashing based techniques cannot be used for finding similar images, since even minimal alterations, such as cropping and resizing, would make different copies of the same image untraceable. Image similarity techniques are desired to retrieve semantically similar images within a database based on a query image. The task of image similarity can be divided into two sub-parts : describing image content in terms of visual features and studying efficient similarity functions over the features. The features from images can be learned by using deep Convolution Neural Networks. 1.3 Organization of The Report This chapter provides a background of the topics covered in this report. The introduction explains the problem statement and its applications in various domains. We review prior works in the domain of graph similarity and various similarity functions that have been proposed.In chapter 3, we provide a analysis of the datasets used for our experiments and also describe the method for converting annotated images to scene graphs. In chapter 4, we discuss the implementation details of the algorithms and discuss the results obtained on those algorithms. And finally in chapter 5, we conclude with limitations and future work. 2 Chapter 2 Review of Prior Works The chapter deals with the review of some of the graph similarity metrics and corresponding algorithms described in prior papers. 2.1 Efficient Graph Similarity Joins with Edit Distance Constraints The subsection provides an overview of [ZXLW12] The graph edit distance between r and s is the minimum number of edit operations that transform r to a graph isomorphic to s.The edit operations consist of insertions, deletions and renaming labels of vertices/edges. The computation of graph edit distance between two graphs is NP-hard. In order to tackle NP-hardness of the problem, the paper proposes a graph similarity join problem with edit distance constraints. Inspired by the idea of q-gram on string similarity, the paper proposes GSimJoin algorithm that defines q- gram on graphs based on path. A path based q-gram in a graph is a simple path of length q. The idea of the edit distance constraints is that an edit operation will only affect a limited number of q-grams. The maximum number of q-grams that can be affected by an edit operation is shown as Dpath = maxuεV (r)|Qur| where Qur denotes the multiset of q-grams that contain vertex u and V (r) denotes the vertex set in graph r. It then defines a count filtering condition for the path-based q-grams that two graphs,r and s must share atleast LBpath = max(|Qr| − τ Dpath(r), |Qs| − τ Dpath(s)) common q-grams if they are within graph edit distance τ . A pair of graph that satisfies the lower bound condition form a candidate pair. The algorithm

uses prefix filtering to find candidate pairs that satisfy count filtering condition. According to path filtering, let Qr and Qs are q-gram multisets for r and s respectively sorted according to global ordering of the q-gram universe. If |Qr ∩ Qs| ≥ α, then the (Qr − α + 1)-prefix of Qr and the (Qs − α + 1)-prefix of Qs, must have atleast one common q-gram. Rare q-grams are favoured in prefixes in order to achieve small candidate size and fast execution. The algorithm then uses minimum edit filtering and label filtering using mismatching q-gram to further reduce the candidate size. It then invokes expensive graph edit calculation for every candidate pair that survive all the filtering conditions to tell if they are join results. 2.2 Vertex-Edge Overlap (VEO) The subsection provides an overview of Vertex/edge overlap from the work [PPGM10] The idea of similarity for Vertex Edge Overlap is based on the rule that “two graphs are similar if they share many vertices and edges”. Graph edit distance and Jaccard Index are two main ways of computing this kind of similarity. We discussed graph edit distance in the previous section. Jaccard index is defined as the intersection of vertices/edges divided 4 by the union of vertices/edges. The similarity function for two graphs G and G′ is defined as simV EO(G, G′) = 2 |V ∩ V ′| + |E ∩ E′| |V | + |V ′| + |E| + |E′| The time complexity for algorithm implementing simV EO is O(|V | + |V ′| + |E| + |E′|).The algorithm implementing this scheme is called Vertex/Edge Overlap algorithm. 2.3 Conclusion This chapter provides details of some of the existing graph similarity metrics and algo- rithms. In the next chapter, we provide an analysis of the datasets used in our experiments and methods to convert annotated images from those datasets to scene graphs, that will be used as input to the above algorithms. Chapter 3 Dataset & Prepro cessing In this chapter, we provide a description of the datasets used. We also describe the prepro- cessing of datasets to construct scene graphs which will be used as input for our algorithm. We then describe the improvements to base algorithms from next chapter onwards. 3.1 Dataset Description In this section, we describe the datasets used for running the experiments. We use two annotated image datasets namely Mapillary Vistas Dataset and Visual Gnome Dataset. We provide a brief description of each one of these in the following subsections. 3.1.1 Mapillary Vistas Dataset In this section, we provide a brief description of Mapillary Vistas Dataset [NORBK17] and describe the process of using the dataset. The dataset can be downloaded from https://www.mapillary.com/dataset/vistas. The dataset contains images segmented into training, validation and testing subsets. For our purpose, we use the images in the training subset since this subset contains annotated images. The training subset contains a total of 18000 images with 66 semantic object categories. A brief summary of the dataset is given in Table 3. 1 7 Number of images 18000 Number of object categories 66 Table 3.1: Mapillary Vistas Dataset description Each semantic object class is represented by a specific colour in the image. The colour representing each object class remains constant throughout the dataset. The information regarding the semantic object class and the corresponding colour representing the object class can be found in config.json file, that is downloaded along with the mapillary vistas dataset. A sample annotated image from the dataset is shown in Figure 3.1 Fig. 3.1: Example from Mapillary Vistas Dataset 3.1.2 Visual Genome Dataset In this section, we provide a brief description of Visual Genome Dataset [KZG+16] and describe the process of using the dataset. The dataset can be downloaded from https: //visualgenome.org/api/v0/api_home.html. For our experiments, we use v1.2 of the dataset. The dataset contains 108,077 images. The dataset in addition to the images, contains objects.json and relationships.json. objects.json is structured as a list of json 8 objects, where each element of the list stores the image id and the list of objects associated with the particular image recognised by the image id. relationships.json is also structured as a list of json objects, where each element of the list stores the image id and a list containing all the relationships recognised between the objects in the image. A brief summary of the dataset is given in Table 3.2 Number of images 108077 Number of distinct object classes 5996 Number of distinct relationship classes 1014 Table 3.2: Visual Genome Dataset description The average distribution statistics of objects and relationships across images is summa- rized in Table 3.3. Average number of objects per image 32.34 Average number of relationships per image 16.81 Table 3.3: Average statistics in Visual Genome Dataset The objects and relationship labels are assigned to images manually. The labels, on one side being more reliable since human intelligence is involved instead of machine learning

frameworks, are also prone to human errors. These errors may result in incorrect results in few cases which we will discuss along with the results. The most commonly occurring objects and relationship classes are listed in Table 3.4. Most common object classes Most common Relationship classes man ON person has woman IN building wears sign behind table next to Table 3.4: Commonly occurring classes in Visual Genome Dataset 3.2 Scene Graph Construction In this section, we describe our method to construct scene graphs for the above mentioned datasets. Since the organization and structure of both the datasets are different, we employ different methods to construct scene graphs for each dataset. Scene graph is a simple undirected graph that captures the semantics of the image scene. Each node in the scene graph, represents a specific object in the image. The nodes in the graph are labelled based on the object class it belongs to. An edge between two nodes in a graph points to a relation between the two objects in the image. The edges may be labelled or unlabelled depending on the quality of information that the dataset provides.The following subsections provide an elaborate description of the scene graph construction methods for both the datasets. 3.2.1 Mapillary Vistas Dataset Each image in the dataset can be visualized as a two-dimensional array of pixels, where each pixel stores the RGB colour value at that particular point in image. We run Breadth First Search (BFS) to find continuous regions. A continuous region is a collection of pixels such that for any two pixels in the region, there exists a path from one pixel to another and all pixels on the path have same colour as the two end point pixel cells. According to the definition of node in scene graph, each continuous region in a image forms a single node in the graphs. Since defining nodes using continuous regions will introduce a lot of noise, we only choose continuous regions whose area is atleast 0.01% of the total image area. Thus the total number of nodes in the scene graph is equal to the number of continuous regions in the annotated image. Two nodes in the scene graph share an edge if and only if the corresponding continuous regions share a pixel boundary. Since there is no specific relationship that can be inferred between the two nodes, the edges are unlabelled. However, since unlabelled edges do not impart any useful information in graph similarity metrics, we provide a label to an edge using a function of labels of the corresponding nodes that it connects. Thus, for n number of distinct object classes in the dataset, there are a total of n 2 distinct edge labels possible. ( ) 3.2.2 Visual Genome Dataset The dataset provides objects and relationship labels as json files. Thus constructing scene graph for visual genome dataset requires parsing the json files. The objects.json file contains information required for constructing the nodes in the scene graph and characterizing them to one of the pre-defined object class. The relationships.json file stores information required for adding edges to the scene graph. Each relation in the relationship.json has an associated predicate with it, that provides basis for assigning the edges in the graph to one of the pre- defined relationship class. The information from both the files is merged to construct a scene graph for an image. One thing to note here is that the labels for objects and relations 11 are of data type strings. Thus before constructing the scene graphs, we assign a numeric label to each object and relation class. 3.3 Conclusion In this chapter, we provide a brief analysis of mapillary vistas dataset and visual genome dataset used for our experiments. We also describe the method for conversion of annotated images in both the datasets to scene graphs. For each dataset, we create a single file that stores the scene graphs for all the images in the dataset. This single file, serves as a input to the algorithms that we discuss in the next chapter. In the next chapter, we discuss the graph similarity algorithms and their implementation in detail. Chapter 4 Algorithms In the previous chapter, we described the method for creation of scene graphs. The output of this is a single file containing scene graphs for all the images in a dataset. This file will serve as a input to our algorithm. 4.1 Graph similarity join with edit distance constraints The following section describes in brief the graph similarity join algorithm with edit distance constraints. The idea behind the algorithm is described briefly in Section 2.1. The implementation of the code is borrowed from the paper [ZXLW12]. The algorithm takes as input a set of graphs, r, and maximum permissible edit distance, τ, and returns as output the list of graphs that satisfy the edit distance constraints. The algorithm first uses prefix filtering to implement count filtering condition to get possible candidate pairs. The algorithm then uses minimum edit filtering and local label filtering to estimate the lower bound of the graph edit distance for the candidate pairs

based on mismatching q-grams. If the lower bound for a candidate pair is greater than than the maximum permissible graph edit distance τ , the candidate pair is discarded. The remaining candidate pairs are then subjected to expensive graph edit distance calculation to output the final pairs. 13 4.1.1 Results We run the graph similarity join with edit distance constraints on mapillary vistas dataset. A few results for various graph edit distance threshold are shown in Figure 4.1 (a) τ = 1 (b) τ = 3 Fig. 4.1: Similar image pairs for various τ values 4.2 Vertex Edge Overlap The following section explains the implementation of Vertex Edge Overlap algorithm. The pseudo code for the algorithm in Algorithm 1 The algorithm takes as input two graphs G(V, E) and G′(V ′, E′). The vertex and edge list are sorted according to their label id's. Line 1 of the algorithm finds the intersection of vertex set of the two graphs. Since the vertex list is sorted based on the label id of the vertices, this step takes O(|V | + |V ′|) time steps using two pointer method. Line 2 of 14 Algorithm 1: Vertex Edge Overlap Input: Graph G and G′ with vertices and edges sorted by label Id. Let V and V ′ denote the vertex list and E and E′ denote the edge list for G and G′ respectively Output: Similarity score ∈ [0,1] 1 CV = V ∩ V ′ ; 2 CE = E ∩ E′ ; 3 similarityScore = 2× |V |+|C|VV′||++||CEE|+||E′| the algorithm finds the intersection of edge set of the two graphs. Since the edge list is sorted based on label id of edges, this step takes O(|E| + |E′|) time steps using two pointer method. Finally the similarity score is calculated using the formula as explained in Section 2.2. The overall time complexity for the algorithm is O(|V | + |V ′| + |E| + |E′|). 4.2.1 Results We run the vertex edge overlap algorithm on mapillary vistas dataset and visual genome dataset. A few results on mapillary vistas dataset are shown in Figure 4.2 (a) Similarity Score = 0.86 (b) Similarity Score = 0.83 Fig. 4.2: VEO result on mapillary vistas dataset 15 A few results of vertex edge overlap algorithm on visual genome dataset are shown in Figure 4.3 (a) Similarity Score = 1.0 (b) Similarity Score = 0. 91 Fig. 4. 3: VEO result on visual genome dataset 4.3 Vertex Edge Overlap with partial matching In vertex edge overlap algorithm, only exact matching vertices and edges contribute in the final similarity score. However, when comparing scene graphs of two images, partial match score of vertices must also be incorporated while calculating the final similarity score. This can be understood by the fact that two different object classes may represent semanti- cally similar objects. For instance, person and man, and pants and trousers represents semantically similar objects. We now elaborate the complete algorithm along with our modification. The algorithm is summarized in Algorithm 2 The algorithm takes as input two graphs G(V, E) and G′(V ′, E′). The vertex and edge list are sorted according to their label id's. Line 1-2 of the algorithm finds the intersection 16 Algorithm 2: Vertex Edge Overlap with partial matching Input: Graph G and G′ with vertices and edges sorted by label Id. Let V and V′ denote the vertex list and E and E′ denote the edge list for G and G′ respectively Output: Similarity score ∈ [0,1] 1 CV = V ∩ V′ ; 2 CE = E ∩ E′ ; 3 NCV G = V − CV ; 4 NCV G′ = V′ − CV ; 5 Construct 2D-array costMatrix of size |NCV G| × |NCV G′| where costMatrix[i][j] denotes the partial similarity score between vertex labels NCV G[i] and NCV G′[j] 6 maxWeightedMatching = 0 7 for i = 1 to P do 8 Randomly shuffle the rows of costMatrix 9 matchingScore = Starting from 1st row, greedily choose the maximum weighted matching score ; 10 maxWeightedMatching = max(maxWeightedMatching, matchingScore) 11 end 12 similarityScore = 2× |CV |+|CE|V|+|+m|Vax′|W+|eEig|+ht|Eed′M|atching ; vertex and edge list and is implemented using the tow pointer method. Line 3-4 finds the mismatched vertex and edge labels in the two graphs. This is again implemented using two pointer method and thus takes time linear in the number of vertices and edges. Line 5-11 implements the partial matching of vertices. Line 5 construct a 2D-array costmatrix of size |NCV G| × |NCV G′|. The element at indexed at (i, j) in costM atrix, is the partial similarity score between vertex label NCV G[i] and NCV G′[j]. To calculate the partial similarity score between vertex labels, which are originally strings as described in Section 3.2. 2, we use binary word embedding from fasttext [MGB+18]. Using fasttext, each label is then transformed into a 300-dimensional vector. The cosine similarity between the vectors of the corresponding labels are used as partial similarity score between the vertex labels. The algorithm then uses a greedy approach to calculate the maximum weighted matching. Line 6-11 implements this part of the algorithm. At each iteration, we randomly shuffle the rows of the costM atrix, and then starting from first row, greedily calculate the maximum possible weighted matching in the given permutation.

The similarity score is then calculated using the similar formula as in simple Vertex Edge Overlap, except the fact that partial similarity score between vertices is also taken into account. 4.3.1 Additional Improvements The greedy maximum weighted matching approach returns values close to the maximum weighted matching score in most cases but it does not return the optimal maximum weighted matching score in some of the cases. The problem of maximum weighted matching is implemented by the Hungarian Algorithm [Kuh55]. We tried using Hungarian algorithm in place of greedy approach, but since the time complexity of Hungarian Algorithm is O(V 3), the VEO with partial matching takes around 300 days to obtain results for pair- wise similarity score for all possible image pairs in visual genome dataset. Thus, we only stick to greedy approach that yields results close to the optimal value and has a time complexity of O(V 2). 4.3.2 Results We run vertex edge overlap with partial matching on visual genome dataset. Top 5 image pairs with highest similarity scores are shown in Figure 4.4 4.4 Conclusion In this chapter, we discuss the implementation details of various algorithms used and results obtained by running these algorithms on the datasets. In the next chapter, we discuss the limitations of the datasets and algorithms and discuss the future work that can pave way for a finer similarity function. 18 (a) Similarity Score = 1.0 (b) Similarity Score = 1.0 (c) Similarity Score = 0.933 (d) Similarity Score = 0.931 (e) Similarity Score = 0.917 Fig. 4.4: Top 5 results for VEO with partial matching on visual genome dataset Chapter 5 Limitations & Future Work 5.1 Limitations There are few limitations of the dataset that limit the quality of results that are obtained. The mapillary vistas dataset, does not contain relationship annotations between objects which leads to semantically distinct image pairs being labelled as similar by the algorithm. The visual genome dataset, which has human annotated images, has incorrectly labelled images due to human error. Since the algorithms uses these labels to characterize the images as similar, this results in quite distinct images being labelled as similar. One such example of incorrect annotation of images that leads to high similarity score is shown in Figure 5.1. In the second sub figure, the aeroplane is incorrectly labelled as train, which results in high similarity score between the image pairs. A better quality dataset is thus required to improve and assess the quality of results obtained by the algorithms described in previous chapter. 5.2 Future Work The limitations with datasets can be resolved by captioning images using deep neural net- works like DenseCap : Fully Convolutional Localization Networks for Dense Captioning 21 (a) Similarity Score = 0.92 Fig. 5.1: Incorrect image annotation in visual genome dataset [JKL15]. Natural language processing can be applied to these machine generated captions to provide suitable labels to the image objects and their corresponding relationships. Moreover, simple Vertex Edge Overlap and Vertex Edge Overlap with partial matching imparts equal weight to every label type during calculation of similarity score. However, certain label classes occur more frequently across images than others and thus must be given smaller weight in computation of similarity score. References [JKL15] [JKS+ 15] [Kuh55] [KZG+ 16] [MGB+ 18] [ML19] Justin Johnson, Andrej Karpathy, and Fei-Fei Li. Densecap: Fully convo- lutional localization networks for dense captioning. CoRR, abs/1511.07571, 2015. J. Johnson, R. Krishna, M. Stark, L. Li, D. A. Shamma, M. S. Bernstein, and L. Fei-Fei. Image retrieval using scene graphs. In 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 3668–3678, 2015. H. W. Kuhn. The hungarian method for the assignment problem. Naval Research Logistics Quarterly, 2(1-2):83–97, 1955. Ranjay Krishna, Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen, Yannis Kalantidis, Li-Jia Li, David A Shamma, Michael Bernstein, and Li Fei-Fei. Visual genome: Connecting language and vision using crowdsourced dense image annotations. 2016. Tomas Mikolov, Edouard Grave, Piotr Bojanowski, Christian Puhrsch, and Armand Joulin. Advances in pre-training distributed word representations. In Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018), 2018. Lia Morra and Fabrizio Lamberti. Benchmarking unsupervised near-duplicate image detection. Expert Systems with Applications, 135:313 – 326, 2019. 23 [NORBK17] Gerhard Neuhold, Tobias Ollmann, Samuel Rota Bulò, and Peter Kontschieder. The mapillary vistas dataset for semantic understanding of street scenes. In International Conference on Computer Vision (ICCV), 2017. [PPGM10] [ST20] [ZXLW12] Ali Dasdan Panagiotis Papadimitriou and Hector Garcia-Molina. Web graph similarity for anomaly detection.

Journal of Internet Services and Applica- tions, 1:19 – 30, 2010. B. Schroeder and S. Tripathi. Structured query-based image retrieval using scene graphs. In 2020 IEEE/CVF Conference on Computer Vision and Pat- tern Recognition Workshops (CVPRW), pages 680–684, 2020. X. Zhao, C. Xiao, X. Lin, and W. Wang. Efficient graph similarity joins with edit distance constraints. In 2012 IEEE 28th International Conference on Data Engineering, pages 834–845, April 2012. 3 5 6 9 10 12 17 19 20 22 24