Spring 5-23-2021

On Stability and Similarity of Network Embeddings

Apurva Shriniwas Mulay
Syracuse University

ABSTRACT

Machine Learning on graphs has become an active research area due to the prevailing graph-structured data in the real world. Many real-world applications can be modeled with graphs. Modern application domains include web-scale social networks [26], recommender systems, knowledge graphs, and biological or protein networks. However, there are various challenges. First, the graphs generated from such applications are often large. Moreover, in some scenarios, the complete graph is not available, e.g., for privacy reasons. Thus, it becomes impractical to perform network analysis or compute various graph measures. Hence, graph sampling becomes an important task.

Sampling is often the first step to handle any type of massive data. The same applies to graphs as well, which leads to many graph sampling techniques. Sampling Techniques include *Node-based* (e.g., Random Node Sampling), *Edge-based* (e.g., Random Edge Sampling) and *Traversal-based* (e.g., Random Walk Sampling). Graphs are often analyzed by first *embedding* (i.e., representing) them in some matrix/vector form with some number of dimensions. Various *graph embedding* methods have been developed to convert raw graph data into high dimensional vectors while preserving intrinsic graph properties [3]. The embedding methods focus on the node-level, edge- level [28], a hybrid, or at the graph level. This thesis focuses on graph-level embeddings which allows calculating similarity between two graphs.

With the knowledge of embedding and sampling methods, the natural questions to ask are: 1) What is a good sampling size to ensure embeddings are similar enough to that of the original graph? 2) Do results depend on the sampling method? 3) Do they depend on the embedding method? 4) As we have embeddings, can we find some similarity between the original graph and sample? 5) How do we decide if the sample is good or not? How do we decide if the embedding is good or not? Essentially, if we have an embedding method and a sampling strategy, can we find the smallest sampling size that will give an ε-similar embedding to that of the original graph? We will try to answer the above questions in the thesis and give a new perspective on graph sampling.

The experiments are conducted on graphs with thousands of edges and nodes. The datasets include graphs from social networks, autonomous systems, peer-to-peer networks, and collaboration networks. Two sampling methods are targeted namely - Random Node Sampling, and Random Edge Sampling. Euclidean distance is used as a similarity metric. Experiments are carried out on Graph2vec, and Spectral Features(SF) graph embedding methods. Univariate analysis is performed to decide a minimum sample which gives, e.g., 40% minimum sample for 80% similarity. We also design a Regression model which predicts similarity for a given sampling size and graph properties. Finally, we analyze the *stability* of the embedding methods, where we find that that e.g., Graph2Vec is a stable embedding method.

On Stability and Similarity of Network Embeddings

by

Apurva Mulay

Bachelor of Engineering, Pune University, 2015

Thesis

Submitted in partial fulfillment of the requirement for the degree of

Master of Science, Computer Science

Syracuse University

May 2021

Acknowledgements

My graduate school journey was not an easy one but would not have been fun without help and support from others.

First, I would like to thank my advisor Prof. Reza Zafarani, without whom this thesis was impossible. I got interested in data science and research when I took the "Introduction to data science" course under him. He always encouraged me to push myself that opened my mind to work on stuff I thought would be difficult for me. This thesis is a part of it. He strove to keep the thesis as my original work but steered me in the right direction when required. He not only guided me in research but also helped me stay motivated during the pandemic. I will always be grateful to him for bringing out the best in me.

Coming from an industry Software Engineering background into Data Science research required a shift in the thinking process. Professor Reza and all my lab mates from Data Lab helped me develop it throughout the thesis. Special thanks to Shengmin for answering all kinds of questions and to help me think through the research questions. Thank you to Xinyi as well for showing me the approach for reading research papers.

I would like to thank my committee members Professor Sara Eftekharnejad, Professor Edmund Yu, and Professor Senem Velipasalar for their time and comments.

Finally, I would like to thank my family and friends for their continuous support and for cheering me up during my low times.

Contents

List of Tables

List of Figures

Page

Page

Page

Chapter 1

INTRODUCTION

We live in a world where networks are ingrained in our daily life. We form a network when we connect to others on social media or send an email or message to others, forming a *friendship network*. We create a post or interact with other posts, thus forming a network of social media posts. An example of such a network is a retweet network on Twitter or page-to-page network on Facebook. When we decide to travel, the first thing we check is Google Maps! Google maps essentially provides a network connecting roads and using different transportation, forming *transportation networks*. When we plan a vacation, we check flights from where we live to the travel destination. Many such flights from one's home to the same or different destinations form *airline networks*. To navigate the Web, we use Google search, where the idea behind its fast paced search is by ranking a *Knowledge graph*. When we order products from Amazon or groceries from Instacart, we are forming *product co-purchasing networks*. The co-purchasing network is created when we frequently buy multiple products together. A well-known example of products bought together is bread and butter. Finally, the electricity and water we have in our homes are part of the electrical and water networks for our area or state. When thinking of research papers, citing other papers forms *citation networks*. As we can see, networks are everywhere. Formally, networks are represented as graphs. Graphs consist of nodes that represent entities and edges that represent links between entities. With an underlying representation of networks, many researchers have started focusing on analyzing these graphs. One simple example we use in our daily lives is checking the shortest path on google maps to reach the destination. The graphs are also extremely helpful in understanding the propagation of information on social media. Such analyses help find out viral posts, combat fake news, and broadcast important information quickly.

While graphs are present in most aspects of our lives, they are becoming larger and larger. Hence, Machine Learning on large graphs has become a vital research area. However, the larger

graphs will come with a caveat of space and computing limitations. Thus, it often becomes impractical to perform machine learning techniques and analysis on the whole network. The solution to this problem often involves exploring sampling. Having larger graphs is one of the motivations for sampling. We direct interested readers to the excellent graph sampling survey by Hu et al. [8], which provides more inspiration for sampling. Next, we will talk about the motivations, contributions, and road map for this thesis in detail.

1.1 Motivation

The main motivations behind this thesis can be summarized in terms of (1) lack of data; (2) surveying hidden populations; (3) graph sparsification; (4) visualization; (5) reducing test cost; and to (6) create a map between various sampling techniques.

1. Lack of data. It is often practically impossible to collect all of a network, e.g., a social network such as Twitter. The easiest way is to choose a few users (i.e., nodes) randomly and crawl them. The Twitter's search API follows a similar process to retrieve tweets, user information, etc. This is essentially *random node sampling*.

2. Survey hidden populations. In sociology, there is a need to survey hidden populations. Hidden populations have two characteristics. First, no sampling frame exists as boundaries are unknown. Second, there are privacy concerns as membership involves illegal or stigmatized behaviors [4]. Thus, it is impossible to sample from whole population. The usual approach is to start from a small sample and expand the sample based on some prior knowledge. This approach is known as *Snow Ball Sampling* [5] and *Respondent-Driven Sampling* [7]. In this thesis, we do not discuss these two sampling methods, but the techniques presented can be easily extended to those in future. Surveying hidden population is one major motivation for sampling.

3. Graph Sparsification. Real world graphs are very large in size which makes it difficult to process or manipulate them. This calls for *Graph Sparsification*. Graph Sparsification is classical problem in computer science theory and involves stringent mathematical constraints on transformation of graphs that ensure major properties of the graphs are preserved.

4. Visualization. With the amount of data being generated, there is often a difficulty to

visualize large graphs as they do not fit in the computer screen. One option is to adjust the resolution which often leads to edges crossing and being too cluttered. Kurant et al. [10] show how graph sampling can be used to obtain a coarse-grained topology proper for visualization, i.e., a *category-graph* estimation. Another approach is to perform *Dimensionality Reduction* and *Graph Embedding*. We discuss two graph embedding techniques in this thesis for sampling graphs.

5. Reducing test cost. In biochemical research, Protein Interaction Network [6] is often studied. Accurate testing of interaction between all neighboring proteins is often costly. A approximate solution here is to test on sampled edges.

6. Mapping between various sampling methods. Different researchers sample graphs using various methods. One use case is Graph sparsification which includes edge and vertex sparsification. It is difficult to compare two analyses when both analyses use different sampling methods. Thus, our work can provide research directions of the mapping between minimum sampling sizes across different samples.

1.2 Research Questions

1. What is a *good sampling strategy* to get embeddings similar to original graph?

2. What is a *good embedding method* that gives higher stability?

3. If we have an embedding method and a sampling strategy, what is the *smallest sampling size* that will give an embedding similar enough to that of the original graph?

4. If we know the sample size, embedding method and a sampling strategy, can we *predict the similarity* of the sample to that of original graph?

1.3 Contributions

In this thesis, we make the following contributions:

Identifying minimum sampling size. We start with identifying the minimum sampling size with which the graph needs to be sampled to get an ε-similarity to the original graph, for a given embedding and sampling method. The ε varies with embedding and sampling methods. We start

with social networks data and then including networks from multiple domains to generalize the results.

Regression Model to predict similarity. Identifying the minimum sample size calculation is merely based on the similarity between embeddings with respect to sampling sizes. However, this would not be specific to graphs. So we build a regression model which, given a sampling size and graph properties, predicts similarity. The model is again domain-specific for social networks and is then generalized to the autonomous systems networks, collaboration networks, peer-to-peer networks.

Stability of Embeddings. We can decide which sampling method to chose, but how do we know which embedding method is better? Hence, we decided to analyze the *stability* of the embedding methods. We decide stability using the R-Squared measure of the regression model defined earlier. The output of regression model is similarity which is nothing but euclidean distance between sample and original graph bounded between 0 and 1. Thus, the higher the R-Squared, the more stable the model, as ultimately stability would mean how much predictable the result is. Note that while a higher R-Squared value indicates higher stability, the specific value of R-Squared does not capture much information on stability and is method-dependent.

1.4 Roadmap

The remainder of the thesis is organized as follows. We first discuss related work and summarize related research in Chapter 2. Chapter 3 details our approach, datasets used, our exploratory data analysis, and graph properties used in our approach. In Chapter 4, we discuss the experiments on similarity and stability analysis. Chapter 5 concludes our results and provides future directions.

Chapter 2

RELATED WORK

This chapter provides a summary of the papers related to graph sampling and embedding, and the related research to the research pursued in this thesis.

The work of Leskovec et al. [11] answers similar questions to ours about the best sampling method and minimum sampling size; however, not within the context of graph embedding. This work considers static directed graphs and graphs with temporal (time) information. The authors define two goals on sampling namely: Back-in-time sampling and Scale-down sampling. The back-in-time goal includes traveling back in time and taking the static snapshot of Graph G_t at that time instant t. The goal is to find the sample S from the original graph G which is similar to snapshot G_t. Scale-down goal consist of finding sample graph S with n' nodes taken from Graph G having n nodes such that $n >> n'$. In simple words, we have Graph G and the size of sample s' (say), then we need to find sampled graph G_s with size s' which has properties similar to Graph G. Next paragraph talks about criteria for evaluation, experiments, and conclusion for static and temporal evolution of graphs considered in this work.

The similarity for static and dynamic graphs is based on similarity of 14 distributions. Criteria for static graphs include the distribution of in-degrees, out-degrees, sizes of the weakly connected and strongly connected components, hop plot, hop plot of the largest weakly connected component, first left singular vector of graph adjacency matrix versus the rank, singular values of graph adjacency matrix versus the rank and clustering coefficient. Criteria for dynamic graphs are measured on the sequence of graphs over time and include Densification Power Law diameter, normalized size of the largest connected component, and average clustering coefficient. Kolmogorov-Smirnov D-statistic is used to compare distribution of the graph properties in sample and the original graph. Their work concludes that there is no single perfect answer to graph sampling and best sampling depends on properties and application, but exploration sampling performs

best overall. Additionally, they conclude that node and exploration-based sampling methods yield good results for 50% sampling while back-in-time gives good samples for 15% sampling using forest fire sampling method.

Morstatter et al. [14] perform a similar analysis for figuring out a good sample of Twitter's streaming API. This work is specific to Twitter APIs while our work generalizes results to all types of graphs. They compare data from twitter's sampled streaming API and the ground truth: the *Firehose*. Sampled streaming API returns limited data while Firehose has unlimited data. The dataset collected is with the same parameters. The analysis is carried out from various perspectives and considers dataset coverage, statistical differences, topical analysis, hashtag analysis, network measures at node and network level, and geographical measures. Network measures are of interest to us as it is related to the thesis. Network measures are calculated for the unweighted User $\times$ User network. The only graph level property that is considered is clustering coefficient and all the properties considered are at node level such as degree centrality, betweenness centrality, and potential reach. Inversely, we consider graph level properties and refrain from using any node level properties. The reason being averaging node level properties to graph level in terms of embedding space comes with numerous limitations. This work does not use any context of embedding and only compare values of properties in the streaming API and Firehose. This work considers largest connected component similar to ours for analysis. They achieve 50-60% accuracy for one day of Streaming API. The network-level measures reveal the correlation between the network centralization index and the proportion of data covered by the streaming API. The authors conclude that the result of using Streaming API highly depends on the coverage and the type of analysis that a researcher wishes to perform.

The earlier paragraphs have work related to ours, however, our methodology differs from them in certain aspects. The next paragraphs will talk about the work that we apply practically in this thesis.

Our work directly relates to graph sampling and the sampling approaches discussed in this work. Pili Hu and Wing Cheong Lau [8] discuss different graph sampling approaches and various

classical as well as advanced properties of graphs. The authors present a taxonomy of graph sampling strategies based on context. Firstly, they categorize sampling methods by objective. Objectives could be getting a representative set of vertices for use cases such as polling on opinion, random street survey, or targeting a hidden population. Secondly, sampling also involves preserving a specific graph property. The use cases consist of estimating the graph properties of the original graph from the sample, or supporting graph algorithms that aim to optimize particular objectives related to specific graph property. Thirdly, some graph generation literature also mentions graph sampling. These methods views graph sampling as generating a graph from the family of graphs. The authors also discuss various categories of graph sampling: vertex sampling, edge sampling, traversal-based sampling, and relationships among them. A traversal-based sampling approach includes various methods such as BFS, DFS, RDS, Random Walk, Metropolis-Hastings Random Walk, Forest Fire, etc. As sampling is often pursued to preserve properties of the graph. Hence, the authors discuss various classical and advanced graph properties that sampling often aims to preserve. Classical properties are network size, degree distribution, average degree, density, average path length, radius, diameter, centrality, etc. We consider these properties in our analysis as well. Advanced properties capture the graph structure in a more detailed way and are used as objectives in graph algorithms. Some examples of advanced graph properties are ratio cut, normalized cut, association, modularity, cohesion, etc.

Our work involves embedding the original graph as well as the samples. We use the embedding by by Narayanan et al. [15] in our work. In other words this work is directly related to our work. The authors first summarize various representations of graphs, which have been used up till now, to address several use cases across domains. The approaches used for representing graphs suffer from quite a few problems, such as loss of generality. This limits the representation approach (using handcrafted features) to be used only for specific domains to solve specific problems. To overcome this limitation, this work proposed a new approach called graph2vec. Graph2vec is a task-agnostic method to create representations or embeddings of graphs in a generalized and data-driven way, using [unsupervised] clustering approach. It is formed using

a neural embedding network. Existing embedding techniques in the space of NLP have been explored. First, word2vec, which is based on the idea that words appearing in the "similar" context should have similar vector representations. An architecture called "skipgram" is used for this. The concept on which skipgram is based, is the maximization of log likelihood, considering the probability of occurrence of context words, given the target word. A context is defined as a fixed number of words surrounding the target word. The second is Negative Sampling, which uses iterations to embed the target word. The third is neural document embedding, which uses a paragraph vector-distributed Bag Of Words, which is also referred to as doc2vec skipgram. Just as doc2vec uses individual words to check if they occur in the same context, graph2vec uses rooted subgraphs to perform the same function. The two reasons why rooted subgraphs are used are: Higher order substructure and Non-linear substructure. The paper describes the way by which rooted subgraphs can be extracted, and how skipgrams with negative sampling can be implemented. The evaluation of the proposed graph2vec model has been performed on five different datasets, namely: MUTAG, PTC, PROTEINS, NCI1, NCI109. SVM algorithm has been used to perform the classification task on these datasets, using each of the embedding methods. The embeddings used for comparison are node2vec, sub2vec, WL Kernel, and Deep WL Kernel. Graph2vec outperforms all other embeddings in the case of MUTAG, PTC, PROTEINS. In the case of NCI1 and NCI109, Deep WL Kernel performs better.

Just as the earlier work, this work is also related directly to our work. In fact, we use the embedding method by de Lara [4] to calculate embeddings for graphs in this thesis. This work is about the simple algorithm for Graph Classification. Graph classification is divided into three categories: graph kernels, sequential methods, and embedding methods that focus on the structure of the graph.

Kernel methods perform pairwise comparisons between graphs of the dataset and apply a classifier on the similarity matrix. These methods aim to construct a kernel to capture useful features and applicable on all graphs.

Sequential methods handle varying sizes of graphs by processing them as a sequence of nodes.

The disadvantage is that it is order-dependent. Embedding methods derive a fixed number of features for each graph that is used as a vector representation for classification. This approach does not depend on the sequence of nodes. The approach in de Lara's work belongs to this category. Next, let us understand the model.

Graph $G = (V, E)$ is an undirected and unweighted graph. The assumption is that the graph is connected. Otherwise, they extract the largest connected component. Matrix A is the adjacency matrix, and D is the diagonal degree matrix. Normalized Laplacian is the crux of this paper. They use the k smallest positive eigenvalues of the normalized Laplacian in ascending order as input to the classifier. If the graph has less than k nodes, right zero paddings are applied to match dimensions. This embedding is called the *Spectral Features (SF)*. Laplacian eigenvalues lie between 0 and 2 and thus eliminates the need for rescaling or preprocessing. The experiments are conducted on standard datasets from biology. Datasets include MUTAG, PTC, Enzymes, Protein Full, DD, and National Cancer Institute. Each dataset is divided into 10 folds using StratifiedKFold and RandomForest classifier is used with balanced class weights. The embedding dimension is set as the average number of nodes for each dataset. The results are compared with those obtained by Earth's Mover's Distance, Pyramid Match, Feature-Based, Dynamic-Based Features, and Stochastic Graphlet embedding. The SF with random forest classifier performs the best overall with efficient computation time.

Finally, Von Luxburg [27] discusses the concept of stability in the context of machine learning model build and selection. We refer to the methodology by Luxburg to decide stability of embeddings in this thesis. The methodology of clustering is taken in this case to explain stability. Initially, the paper focuses on defining the metric and calculations for computing instability. In other words, a stable clustering methodology could roughly be defined as one which produces the same output distribution, given different datasets or different versions of a dataset. One of the parameters used for the clustering model selection is the number of clusters. Different values are chosen for this parameter, and the model outputs corresponding to each of these are compared. The stability scores computed for each of the selections are aggregated using statistical

techniques and normalized to arrive at a standard figure for comparison. Furthermore, the paper also states that this approach is similar to the one defined and could be extended to even other types of clustering algorithms (and not just e.g., k-means). It concludes that the approach needs to be fine-tuned to the particular algorithm or methodology under consideration. We refer the approach from this paper to define stability for embeddings.

2.1 Conclusion

In this section we discussed similar works as well as the methods we used in this thesis. Our work differentiates from the previous work in the aspects of using embeddings and graph properties. To the best of our knowledge, we are the first one calculate minimum sample using embeddings. We use methods Graph2vec [15] and Spectral Features [4] directly in this thesis for calculating embeddings. We refer to concept of stability from Luxburg [27] to analyze stability of embedding methods.

Chapter 3

DATA PREPROCESSING AND OUR APPROACH

This chapter will provide details on our approach to answer questions we initially posed in the introduction section. We first discuss datasets and their statistics, followed by exploratory data analysis and finally input (features) and output (labels) variables for our Machine Learning Model. To get a quick idea, the next paragraph will summarize the complete chapter followed by exact details.

Our datasets consist of graphs from Social Networks, Autonomous Systems, Internet Peer-to-Peer Networks, and Collaboration Networks from SNAP repository [12]. We consider connected graphs. If the graph is not connected, we consider the largest connected component. Firstly, we divide our analysis into two parts. First, we consider only social networks. Second, we add more domains for generalization. Secondly, we create a dataset using graph properties, graph embedding and sampling methods. We focus on two graph embeddings: Graph2vec & Spectral Features, and on Random Node & Random Edge sampling methods. Thirdly, we sample the original graph and create multiple subgraphs using sampling methods mentioned earlier for different sampling sizes (10% to 80%). We get the embeddings for all these samples and that of the original graph based on embedding methods mentioned earlier. Fourth, for each subgraph, we calculate distance between the embeddings of the original graph and that of the sample using Euclidean distance measure. Next, we bound euclidean distances by transforming the similarity values by e^{-x}. Then, we plot the graph of normalized euclidean distance (y-axis) versus sampling size (x-axis) which allows deriving the minimum sampling size that gives a similarity ε times to the original graph. Value ε varies with embedding and sampling methods but our findings show that generally graph2vec gives higher similarity. We hypothesize that one of the reason could be because it is unsupervised and uses neural networks. This analysis will only consider embeddings. However, a graph can also be represented using its properties. Hence, we consider various *graph*

invariants or properties that capture the connectivity of graph efficiently. Lastly, the regression model is used to predict the similarity for a given sampling size and graph properties. We use R-Squared values of the model as a measure of stability for graph embeddings. Figures 3.1 and 3.2 illustrates the approach we followed in this thesis to answer our research questions.

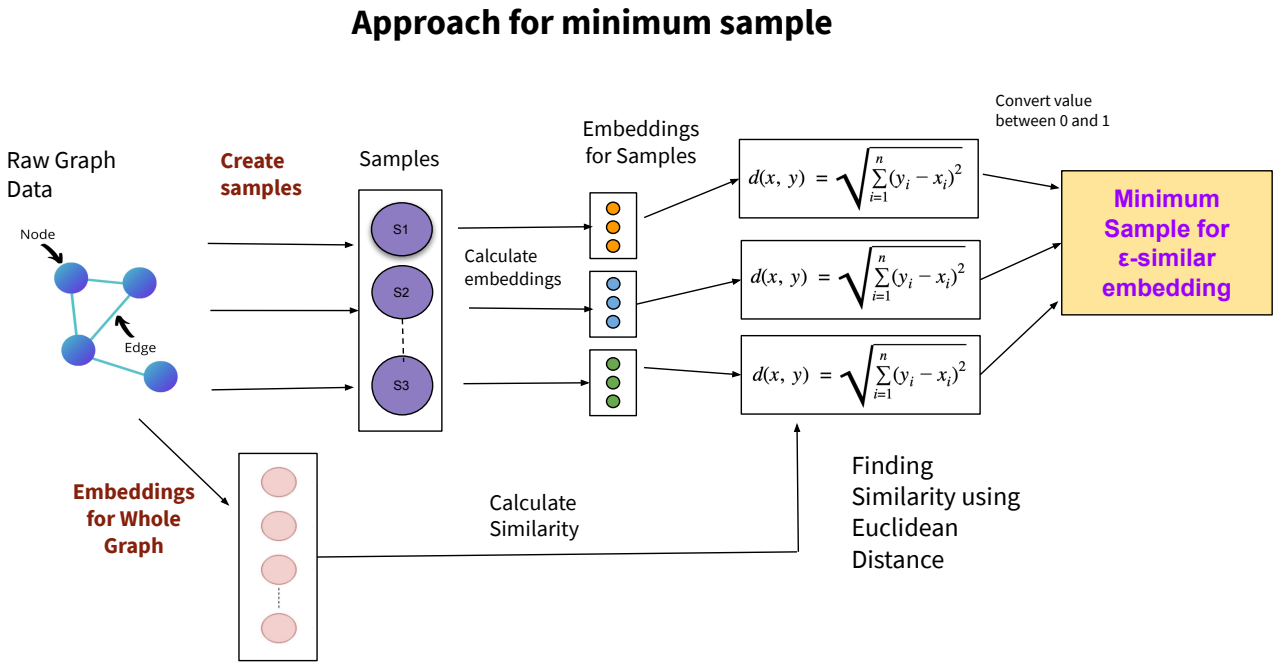


Figure 3.1: Graphical Representation of our approach to find minimum sample

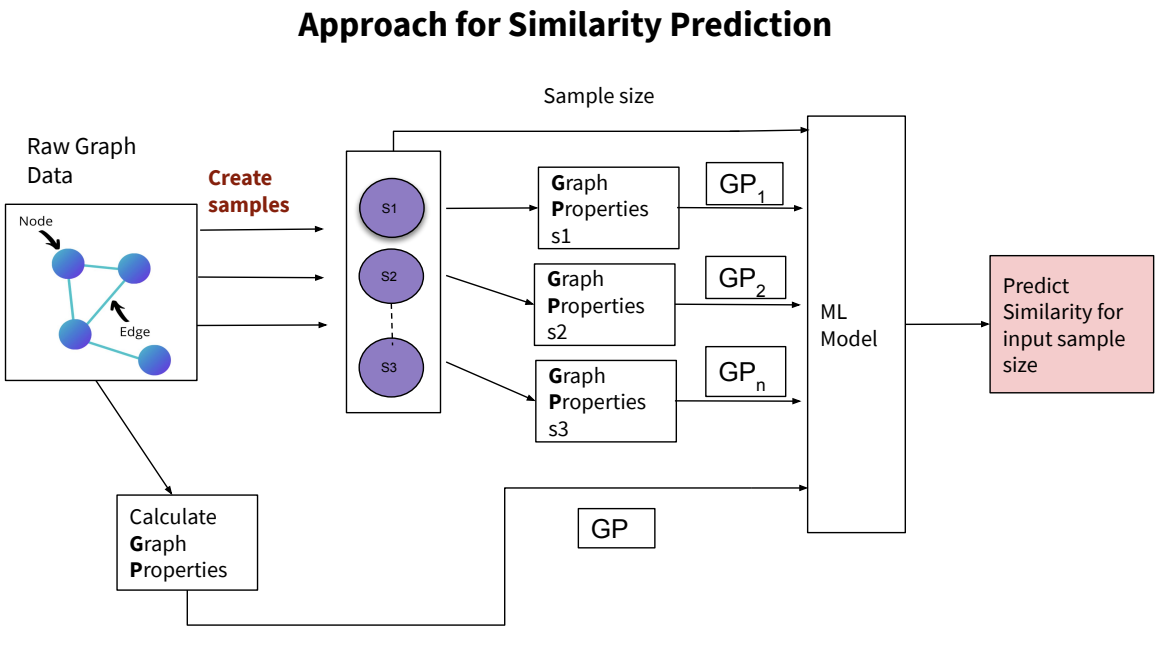


Figure 3.2: Graphical Representation of our Approach. to predict the similarity of input sample size using a regression model

3.1 Datasets

In this thesis we focus on undirected and unweighted network datasets from SNAP [12] large network datasets collection. Some of the datasets are directed but we convert it to undirected during implementation. We consider undirected and unweighted graphs to allow a straightforward analysis.

Table 3.1 summarizes the datasets and their basic properties.

We summarize these datasets next:

(a) **Facebook GEMSEC Dataset**:

This dataset is anonymized and was collected as a part of the GEMSEC [21] paper. It consists of data from Facebook pages collected in November 2017. It represents verified Facebook page

Type	Network	$\|V\| = n$	$\|E\| = m$	Average Shortest Path Length	Average Clustering Coefficient
Collaboration Network	CaGrqc [12]	5,242	14,496	6.0493	0.5296
	Ca-HepTh [12]	9,877	25,998	5.9454	0.4714
	Ca-AstroPh [12]	18,772	198,110	4.1940	0.6306
Social Network	Fb-Athletes [12]	13,866	86,858	4.274	0.27
	Fb-Government [12]	7,057	89,455	3.78	0.410
	Fb-Politician [12]	5,908	41,729	4.664	0.385
	Fb-Public figure [12]	11,565	67,114	4.622	0.179
	Fb-Company [12]	14,113	52,310	5.31	0.2392
	Fb-TV Show [12]	3,892	17262	6.275	0.3737
	Twitch-EN [12]	7,126	35,324	3.677	0.1309
	LastFM-Asia [12]	7,624	27,806	5.232	0.219
Autonomous Systems	As-Jan02-2000 [12]	6,474	13,895	3.705	0.2522
Peer to Peer Network	p2p-Gnutella06 [12]	8,717	31,525	4.571	0.0067

Table 3.1: Dataset Statistics showing basic properties of networks used in the thesis

to page networks of different categories. Pages are the nodes and edges represent mutual likes among them. There are eight different types of categories: Government, New Sites, Athletes, Public Figures, TV Shows, Politician, Artists, and Company Edges. We have not used New Sites and Artist edges due to computation limitations. The statistics of these categories is as mentioned in table 3.1.

(b) **Twitch Social Network Datasets**:

This dataset is provided for MultiScale attributed Node Embedding by Rozemberczki [20]. The dataset consists of Twitch user-user network of gamers who stream in particular language and was collected in May 2018. We focus on the dataset of the English language. Nodes represent the users themselves and the link shows mutual friendships between them.

(c) **Autonomous systems AS-January 2, 2000**:

The dataset was collected from University of Oregon Route Views Project - Online data and report. Autonomous system graphs is graph of routers of internet. The dataset contains daily instances from January 2 2000 onwards. Nodes and edges can get added or deleted overtime.

(d) **Gnutella peer-to-peer network**:

The dataset was first studied by Leskovec [13] and Ripeanu [18]. It consists of snapshots of Gnutella peer-to-peer file sharing network. There are a total of nine snapshots collected in August 2002; we use one snapshot from August 6. Nodes represent hosts in the Gnutella network topology and edges represent connections between the Gnutella hosts. The original dataset is directed but we convert it to undirected to be consistent with other datasets. To make it undirected, we add edge $B \rightarrow A$ for every edge $A \rightarrow B$ where A, B are nodes of the graph.

(e) **High Energy Physics - Phenomenology collaboration network**:

This dataset was first used by Leskovec [13] to study Graph evolution. It covers scientific collaborations between authors of papers submitted to High Energy Physics - Phenomenology category. This is an undirected graph. There is an edge from i to j if author i co-authored a paper with author j. if The data covers papers in the period from January 1993 to April 2003 (124 months).

(f) **Astro Physics collaboration network**:

The dataset was first studied by Leskovec [13] and is collected from arxiv.org for studying graph evolution. It covers scientific collaborations between authors of papers submitted to Astro Physics category. The nodes are authors and edges are undirected. There is an edge between two authors i and j if they have co-authored a paper. Since the edges are undirected, the direction of co-authoring is irrelevant to this thesis. The data covers papers in the period from January 1993 to April 2003 (124 months). In this thesis, we use this dataset as a validation dataset.

(g) **General Relativity and Quantum Cosmology collaboration network**:

This dataset [13] covers scientific collaborations between authors of papers submitted to General Relativity and Quantum Cosmology category. Here, again nodes are authors and edges are collaboration relationship between them.

(h) **LastFM Asia Social Network**:

This dataset was also used by Rozemberczki [19] to validate feathergraph embeddings. It shows social network of LastFM users and was collected in March 2020 using public API. The LastFM users from Asian countries form the nodes and edges show mutual follower relationships among them.

3.2 Exploratory Data Analysis

The datasets used are in the form of edge lists. These datasets needs to be transformed to perform analysis. So we transform the graph using graph embedding methods. A point to note here is that we only consider connected graphs. If the graph is not connected, we extract the largest connected component. First, we explored social networks dataset and then added datasets from other domains to see the performance of the model. The details of these experiments will be discussed in next chapter. Next, we go through the analysis process step by step.

First, we take nine samples of the graphs from 10% to 90% using both Random Node and Random Edge sampling methods. We take 15 samples per sampling size. Thus, we collect 135 samples per dataset. There are a total of eight comma separated values files belonging to datasets from social networks. Please refer table 3.1 and social network rows in the column 'Type' for details about dataset. We refer these comma separated values as datasets. Next, we calculate embeddings for both the original graph and the samples, using Spectral Features and Graph2Vec embedding methods. Now, since we have embeddings of samples and that of the original graph, we calculate similarity using Euclidean Distance measure between two vectors:

$$d(p,q) = \sqrt{\sum_{i=1}^{n} (q_i - p_i)^2} \tag{3.1}$$

where p,q are two vectors, i.e., graph embeddings in our case.

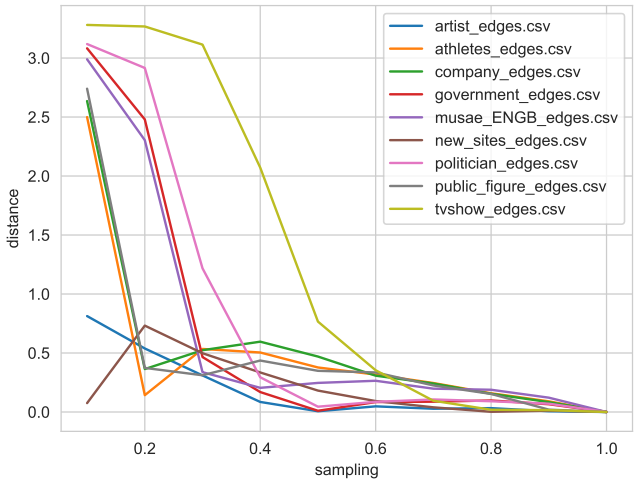


Figure 3.3: Graph of Euclidean Distance versus Sampling Sizes for Random Node Sampling and Graph2Vec Embedding. This includes all social network graphs from SNAP Facebook dataset [12]

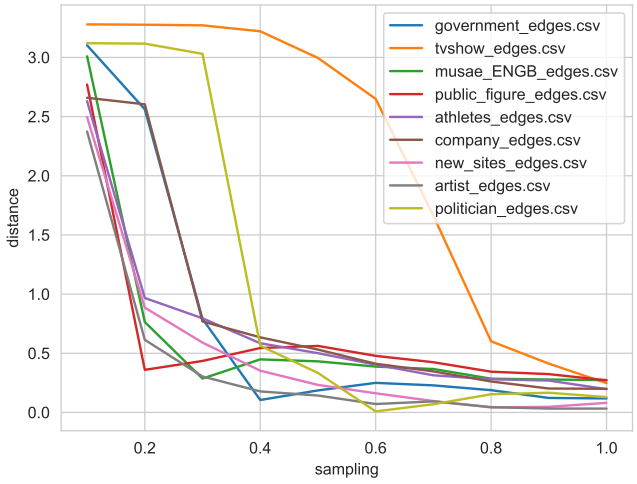


Figure 3.4: Graph of Euclidean Distance versus Sampling Sizes for Random Edge Sampling and Graph2Vec Embedding. This includes all social network graphs from SNAP Facebook dataset [12]

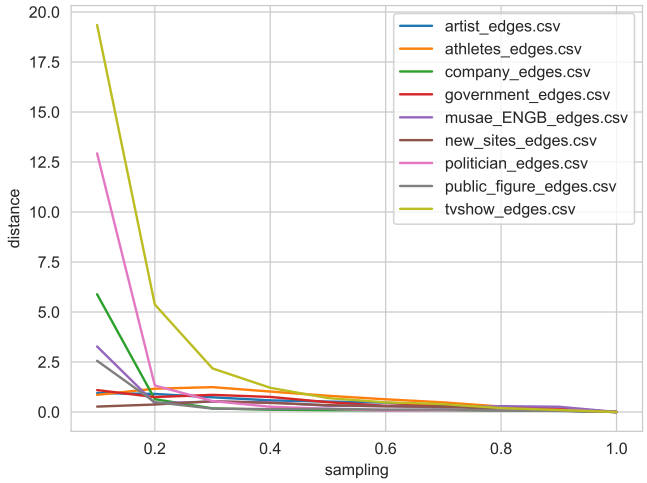


Figure 3.5: Graph of Euclidean Distance versus Sampling Sizes for Random Node Sampling and Spectral Features Embedding.This includes all social network graphs from SNAP Facebook dataset [12]

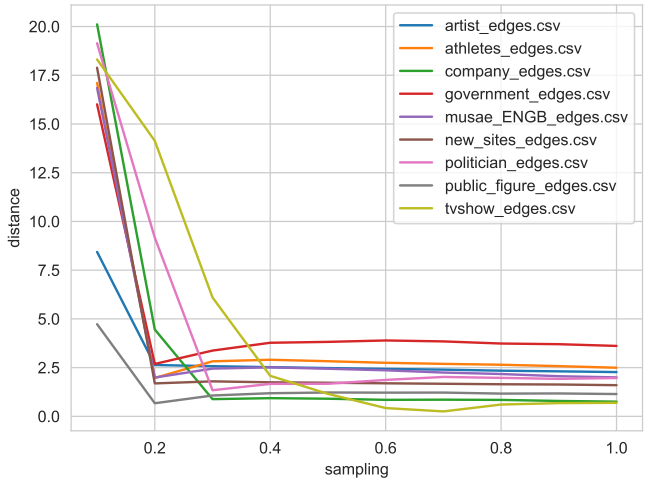


Figure 3.6: Graph of Euclidean Distance versus Sampling Sizes for Random Edge Sampling and Spectral Features Embedding. This includes all social network graphs from SNAP Facebook dataset [12]

Then, we plot this distance versus sampling sizes for all datasets. The plots are shown in

Figures 3.3 to 3.6. The non-linearity of graphs show that there is a potential to find a minimum sample size. Also, from these graphs, it seems that around 40% seems to be a good sample as distances are nearly remaining constant after that. Next, we check the distribution of data, i.e., distribution of Euclidean distances across sampling size for all datasets. The distribution is visualized using box plots of euclidean distances versus datasets. Figures 3.7 to 3.10 show the box plots for Random Node Sampling and Graph2Vec Embedding, Random Edge Sampling and Graph2Vec Embedding, Random Node Sampling and Spectral Features Embedding, and Random Edge Sampling and Spectral Features Embedding. Box plots reveal that there is a large variation between different graphs in the data that needs to be normalized.

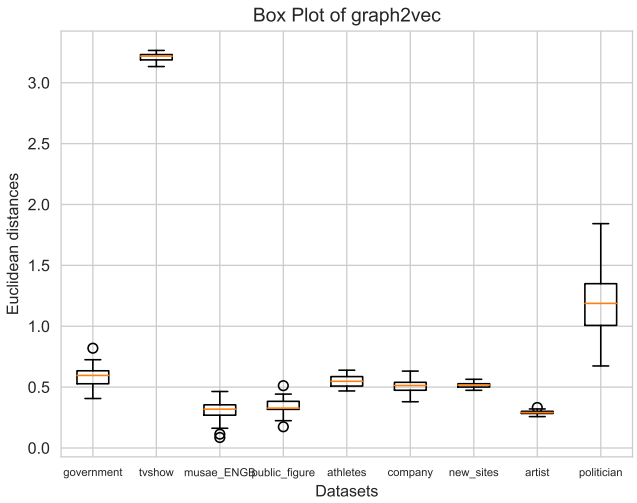


Figure 3.7: Box plot of Euclidean Distance versus all social network graphs for Random Node Sampling and Graph2Vec Embedding. This plot shows distribution of euclidean distances for multiple samples for each graph.

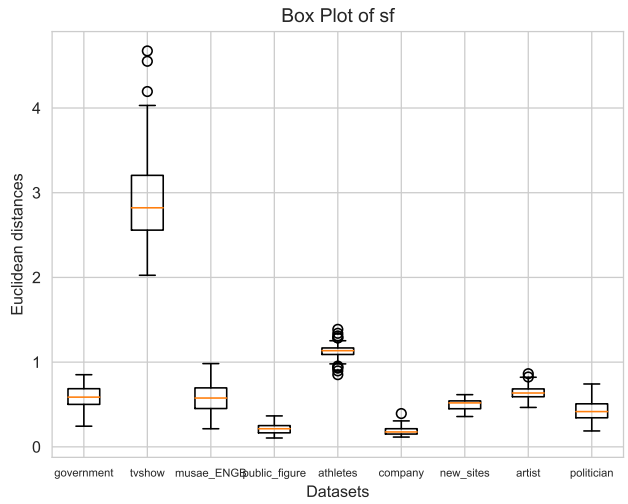


Figure 3.8: Box plot of Euclidean Distance versus all social network graphs for Random Node Sampling and Graph2Vec Embedding.This plot shows distribution of euclidean distances for multiple samples for each graph.

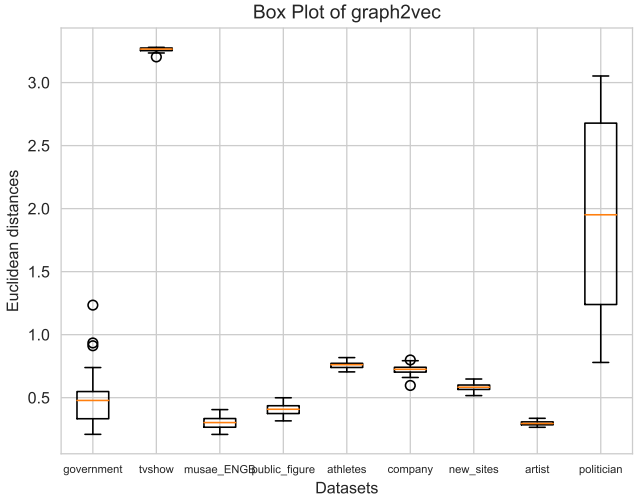


Figure 3.9: Box plot of Euclidean Distance versus all social network graphs for Random Edge Sampling and Graph2Vec Embedding. This plot shows distribution of euclidean distances for multiple samples for each graph.

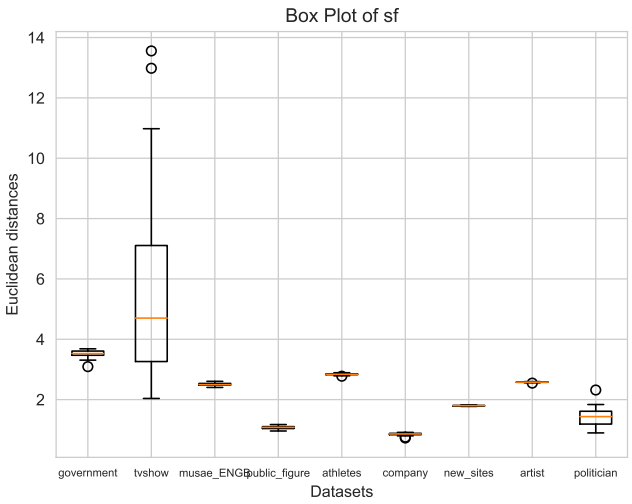


Figure 3.10: Box plot of Euclidean Distance versus all social network graphs for Random Edge Sampling and Spectral Features Embedding. This plot shows distribution of euclidean distances for multiple samples for each graph.

These line plots provide only a two dimensional perspective. Note that one can represent a graph using its properties. Standard properties include average clustering coefficient, number of nodes, number of edges, transitivity, diameter, and density. These properties are also mentioned on SNAP website for respective datasets which is also the source of our utilized datasets [12].

Finally, we also plot a 3D surface to analyze the relation between graph properties, sampling size, and Euclidean distance. We draw the surface plots for every sampling and embedding method. The 3D surfaces show how e.g., clustering coefficient behaves for different sampling and embedding methods. Figure 3.13 and Figure 3.14 show the surface of average clustering coefficient with respect to Graph2Vec and Spectral Features embedding for Random Node Sampling. In both cases, clustering coefficient increases with respect to sampling sizes. Figures 3.11 and 3.12 show the surface of average clustering coefficient with respect to Graph2Vec and Spectral Features embedding for Random Edge Sampling. In both cases, clustering coefficient decreases with respect to sampling size. The decrease is sharp in case of Figure 3.12.

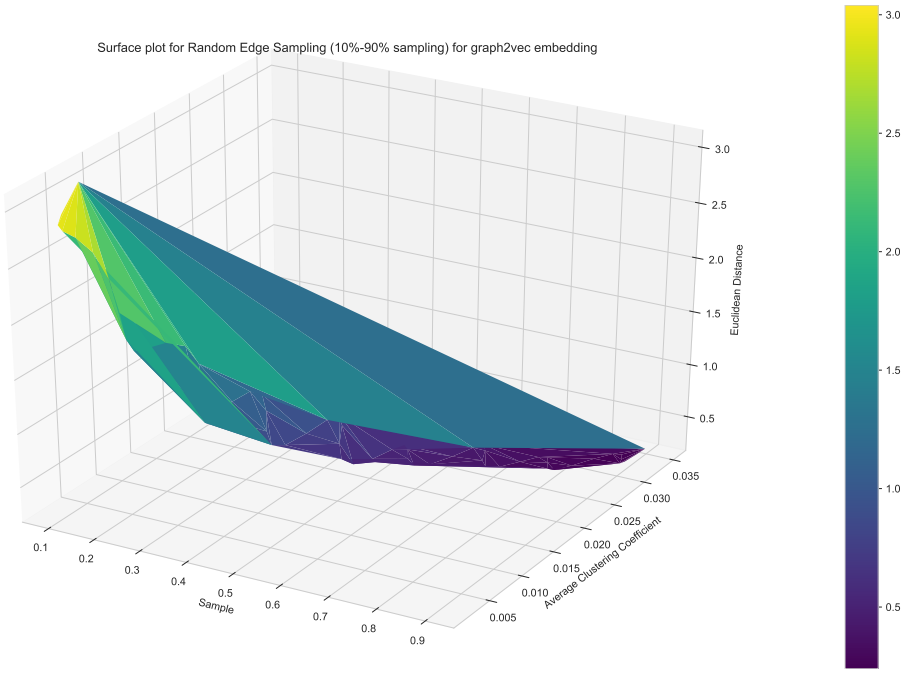


Figure 3.11: Surface plot of Euclidean distance, Sampling Size and Average Clustering Coefficient for Random Edge Sampling and Graph2Vec Embedding

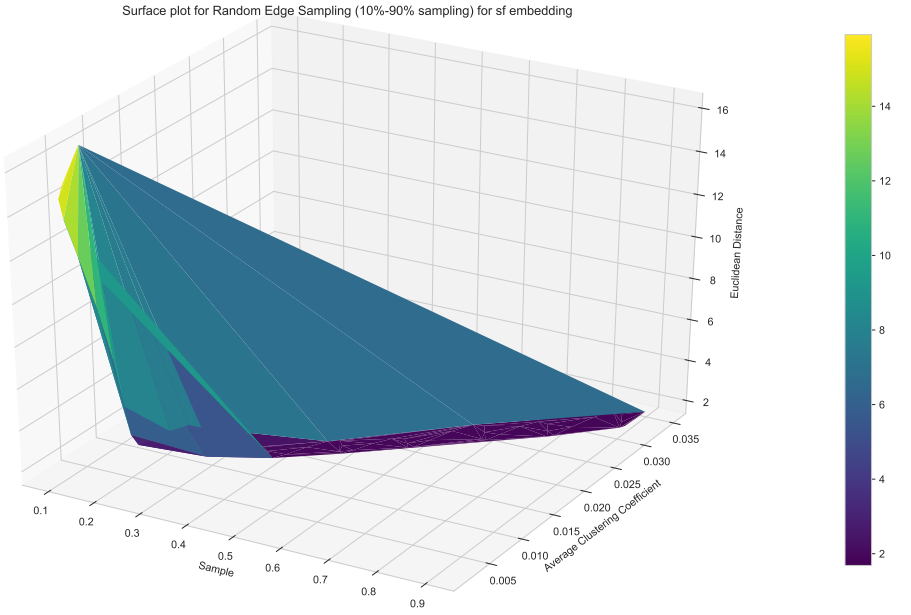


Figure 3.12: Surface plot of Euclidean distance, Sampling Size and Average Clustering Coefficient for Random Edge Sampling and Spectral Features Embedding

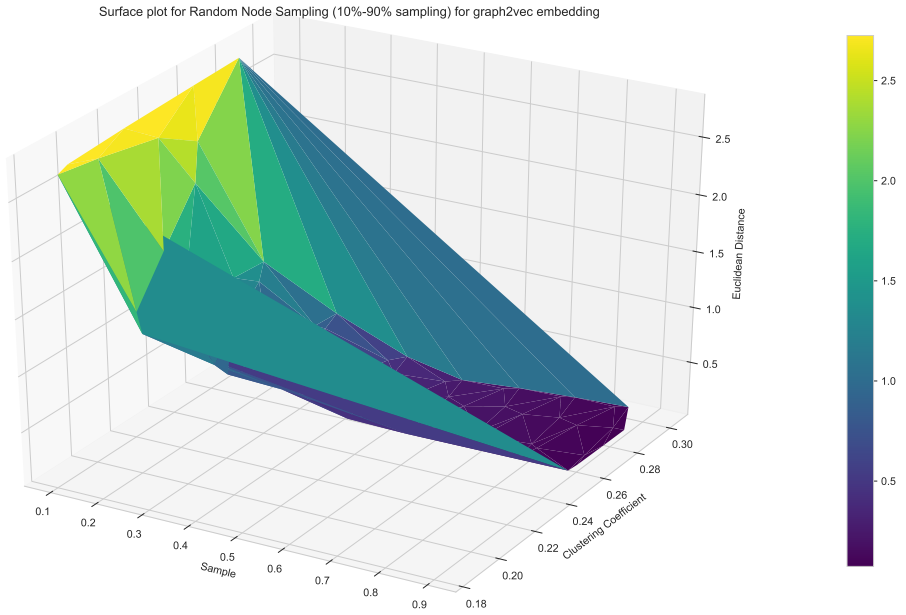


Figure 3.13: Surface plot of Euclidean distance, Sampling Size and Average Clustering Coefficient for Random Node Sampling and Graph2Vec Embedding

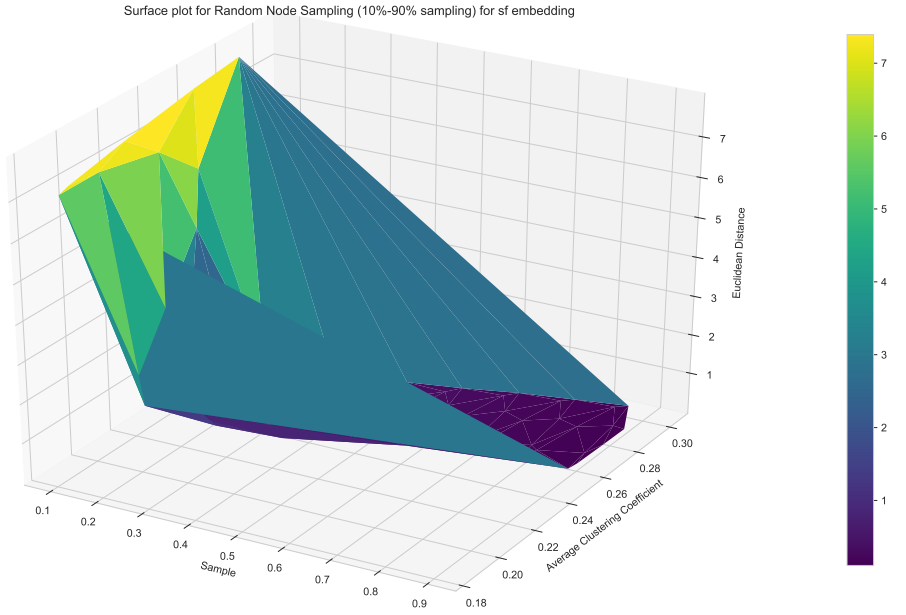


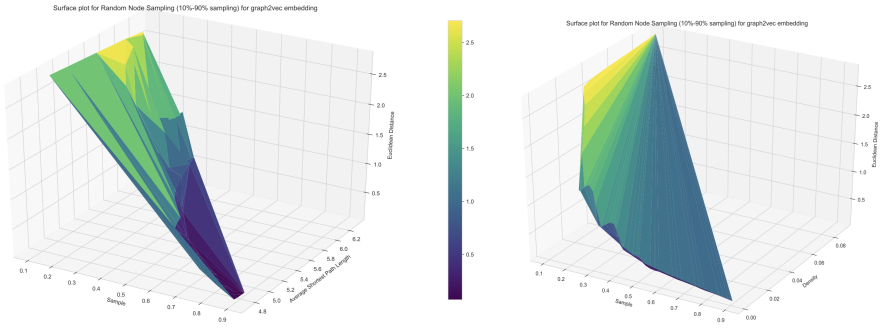
Figure 3.14: Surface plot of Euclidean distance, Sampling Size and Average Clustering Coefficient for Random Node Sampling and Spectral Features Embedding

The figures 3.15 to 3.18 show the surface plot for Average Shortest Path Length, Density, Diameter and Transitivity for Random Node and Edge Sampling, and for Graph2Vec and SF embedding. Our Observations are as follows:

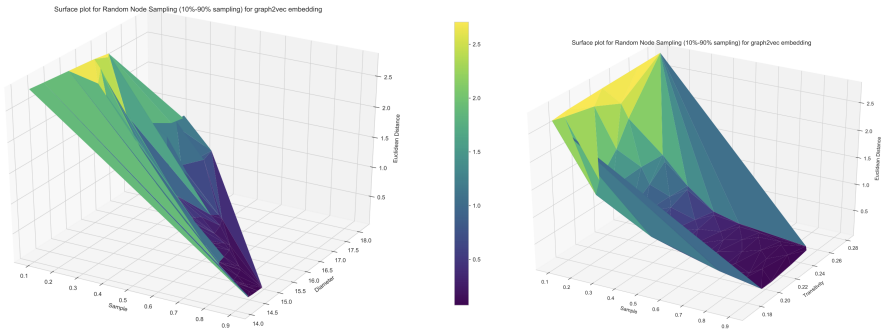
For Graph2vec Embedding: Density decreases with increase in sample size for both random node and random edge sampling methods. Transitivity increases with sample size for node and edge sampling methods. Diameter decreases in random node while it increases in random edge with increase in sample size. Finally, average shortest path length decreases in random node and increases in random edge with increase in sample size.

For Spectral Features Embedding: Density decreases in both random node and random edge sampling. Diameter increases sharply in random edge while it decreases in random node sampling. Transitivity decreases in random node while increases in random edge with increase in sample size. Finally, average shortest path length decreases sharply in random node sampling while it increases in random edge sampling when sample size is increases. As we can see all properties behave different depending on sampling and embedding method.

Random Node Sampling and Graph2Vec Embedding



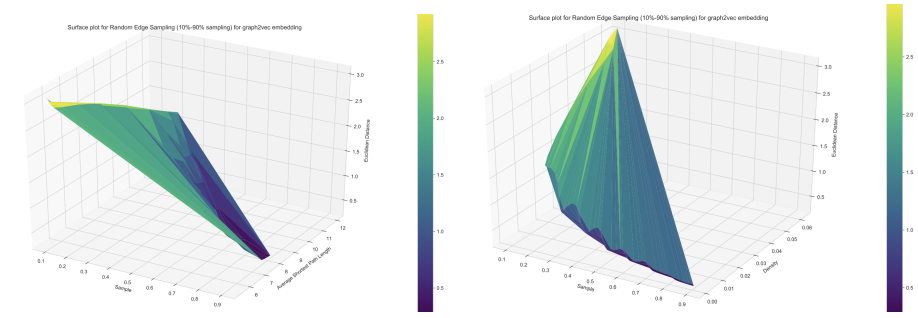
Avg shortest path length Density



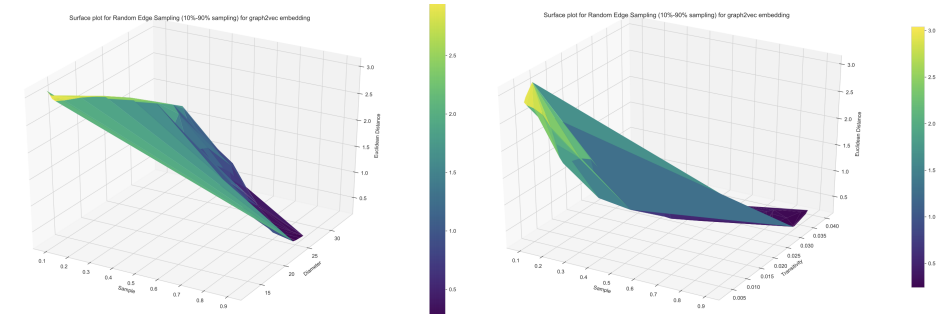
Diameter Transitivity

Figure 3.15: Surface plot of Graph Properties for Random Node Sampling and Graph2Vec Embedding. It includes Avg. Shortest Path Length, Density, Diameter and Transitivity

Random Edge Sampling and Graph2Vec Embedding



Average Shortest Path Length Density



Diameter Transitivity

Figure 3.16: Surface plot of Graph Properties for Random Edge Sampling and Graph2Vec Embedding. It includes Avg. Shortest Path Length, Density, Diameter and Transitivity

Random Node Sampling and Spectral Features Embedding

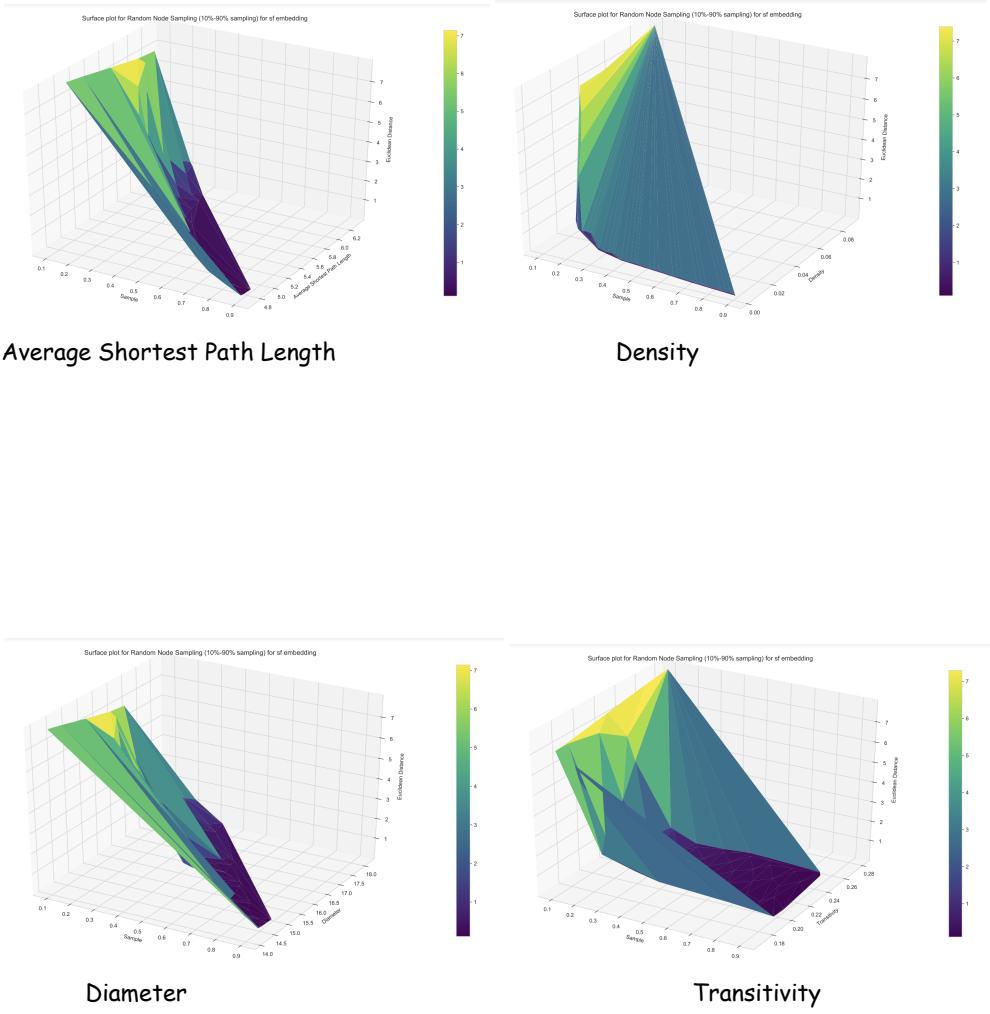


Figure 3.17: Surface plot of Graph Properties for Random Node Sampling and Spectral Features Embedding. It includes Avg. Shortest Path Length, Density, Diameter and Transitivity

Random Edge Sampling and Spectral Features Embedding

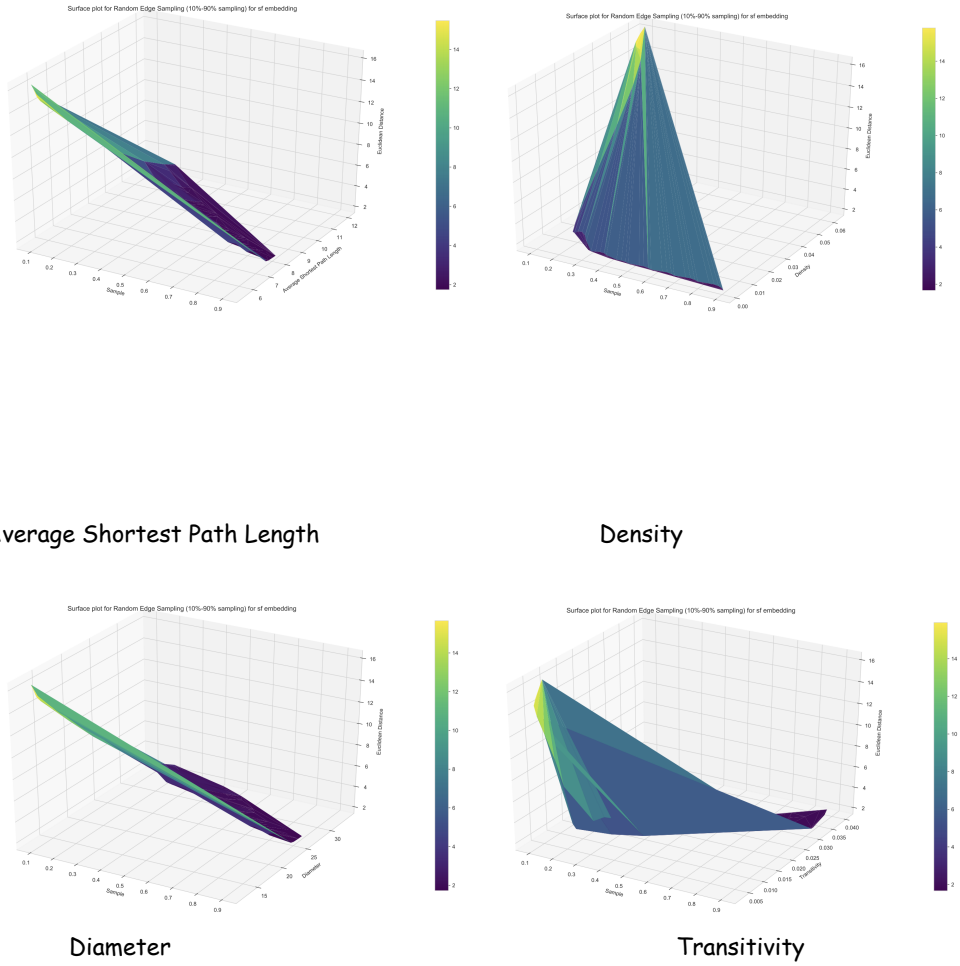


Figure 3.18: Surface plot of Graph Properties for Random Edge Sampling and Spectral Features Embedding. It includes Avg. Shortest Path Length, Density, Diameter and Transitivity

The exploratory data analysis provides us with all the details needed to perform regression with sample and graph properties as independent variables and similarity as dependent variable.

Before delving into the details of regression, we discuss all the graph properties in brief.

3.3 Description of Graph Properties

We will discuss all the graph properties that are used in our experiments. We use networkx and grinpy libraries [**?**] to calculate graph properties. Definitions are taken from wikipedia and the Social Media Mining book [29].

(a) **Clustering coefficient:** A clustering coefficient is a measure of the degree to which nodes in a graph tend to cluster together. There are two versions: local and global. Global gives overall indication of clustering in the network and the local gives an indication of embeddedness of single nodes. Global clustering coefficient can be calculated in an alternate way by averaging local clustering coefficients of all vertices. This alternate method is called as network average clustering coefficient and we use it in this thesis. The network average clustering coefficient is given by:

$$\bar{C} = \frac{1}{n}\sum_{i=1}^{n} C_i, \tag{3.2}$$

where C_i is clustering coefficient for node i.

(b) **Transitivity:** is the fraction of all possible triangles present in graph G. The transitivity is given by:

$$T = 3\frac{\#\text{triangles}}{\#\text{triads}} \tag{3.3}$$

(c) **Average Shortest Path Length:** Average path length is defined as the average number of steps along the shortest paths for all possible pairs of nodes in the network. It is given by formula:

$$l_G = \frac{1}{n \cdot (n-1)} \cdot \sum_{i \neq j} d(v_i, v_j), \tag{3.4}$$

where n is number of vertices in G.

(d) **Density:** The density is the ratio of the number of edges the graph $G(V, E)$ has over maximum number of edges it can have. It can be formalized as:

$$\gamma = \frac{|E|}{\binom{|V|}{2}}. \tag{3.5}$$

(e) **Diameter:** It is the length of longest shortest path between any pair of nodes in the graph. Formally, for a graph G the diameter is defined as:

$$\text{diameter}_G = \max_{(vi,vj)\epsilon V\times V} \quad l_{i,j} \tag{3.6}$$

(f) **Order:** The order of a graph is its number of vertices $|V|$.

(g) **Size:** The size of a graph is its number of edges $|E|$.

(h) **Assortativity:** It measures similarity of connections i.e., correlation between two nodes. One way to capture such correlation is *Assortivity Coefficient*.

The assortativity coefficient is the Pearson correlation coefficient of degree between pairs of linked nodes and is given by:

$$r = \frac{\sum_{jk} jk(e_{jk} - q_j q_k)}{\sigma_q^2}. \tag{3.7}$$

The term q_k is the distribution of the remaining degree and captures edges leaving the node.

(i) **Estrada Index:** It is graph invariant related to eigen values of adjacency matrix. The index was first defined by Ernesto Estrada as a measure of the degree of folding of a protein.

Let $G = (V, E)$ be a graph of size $|V| = n$ and and let $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n \lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n$ be a non-increasing ordering of the eigenvalues of its adjacency matrix A, the Estrada index is defined as:

$$\text{EE}(G) = \sum_{j=1}^{n} e^{\lambda_j}. \tag{3.8}$$

(j) **Circuit Rank:** It is the minimum number of edges that must be removed from the graph to make it a tree. It is equal to the number of independent cycles in the graph. It is computed by using the formula:

$$r = m - n + c \tag{3.9}$$

, where m is the number of edges in the given graph, n is the number of vertices, and c is the number of connected components.

(k) **Girth:** The girth of an undirected graph is the length of a shortest cycle contained in the graph.

(l) **Weiner Index:** It is defined as the sum of the shortest-path distances between each pair of reachable nodes [6]. The Wiener index is named after Harry Wiener, who introduced it in 1947.

(m) **Independence Number:** It is the cardinality of a largest independent set of nodes in the graph.

(n) **Annihilation Number**: as defined in paper [17], the annihilation number is the largest integer k such that the sum of the first k terms of the non-decreasing degree sequence of G is at most the number of edges in G. It is denoted by $\gamma(A_D)$ where D is degree sequence to which process was applied.

(o) **Average Degree:** Average degree is simply the average number of edges per node in the graph. It is total number of edges over total number of nodes:

$$\text{Average Degree} = \frac{TotalEdges}{TotalNodes}. \tag{3.10}$$

(p) **Spectral Moments:** We consider 3 spectral moments second spectral moment (m_2), third spectral moment (m_3) and fourth spectral moment (m_4). First spectral is always 0, hence we eliminate it. To calculate these moments, we refer to code in [9] paper. This authors explain the spectral moments as follows:

The l^{th} spectral moment m_l of a graph G using the spectrum of its random walk transition matrix P,

$$m_l = E(\lambda^l) \tag{3.11}$$

as $\frac{1}{n}\sum_{i=1}^{n}\lambda_i^l = \mathsf{E}(\lambda^l)$

(q) **Radius:** is the minimum graph eccentricity of any vertex in the graph. The eccentricity $\varepsilon(v)$ of vertex v in connected graph is maximum distance between v and any other vertex u of Graph G.

(r) **Residue:** The residue of a graph G is the number of zeros obtained in final sequence of the *Havel-Hakimi process*. Given the degree sequence d of a simple graph, the Havel–Hakimi algorithm is the procedure that iteratively removes a largest term t from the sequence and subtracts 1 from the t largest remaining terms, stopping when a list of zeroes is produced.

We hence have access to the values of the aforementioned graph properties, sampling and embedding methods, sampling sizes, and euclidean distance between embeddings. How can we use this information to answer our questions? The answer to this and all earlier questions will be in the next chapter. In the next chapter we will dive deeper into the experiments we performed and the thought process behind it.

Chapter 4

SIMILARITY AND STABILITY ANALYSIS

This chapter details our experiments on similarity and stability analysis. The analysis includes three parts: First, we study the relationship between similarity and sampling size, i.e., a single variable analysis. This gives us an overall minimum sampling size that gives a similarity nearly equal to original graph. Second, we perform regression to predict similarity for a given sample size and graph properties. We perform this experiment initially on social media graphs followed by utilizing graphs from other domains to generalize the approach. The regression model is created for each pair of embedding and sampling methods. Finally, we perform stability analysis using R-Squared to determine the stable embedding method.

For implementation, we use *Networkx Package* [6] to read edgelists and calculate the graph properties. For sampling methods and extracting samples using those methods, we use *little-balloffur* library [23] by Rozemberczki. To calculate embeddings for graphs, we use the methods from the *karateclub* [22] library again developed by Rozemberczki. We set the default size for embedding vectors, i.e., the embedding dimension is 128.

Before going into details, we first revise sampling methods used in this thesis. Note that we consider sampling with replacement.

Random Node Sampling: The approach, in which, the set of nodes N are selected randomly with uniform probability is called Random Node Sampling. The sampled graph is the graph induced by the set of nodes N.

Random Edge Sampling: Just like random node sampling, Random Edge Sampling selects edges uniformly at random. Meaning that, this method will sample an edge connecting two nodes. The drawback of this method is that it creates very sparsely connected samples. We consider the largest connected component that reduces the probability of getting sparse samples.

Figure 4.1 illustrates the graphical representation for visualization.

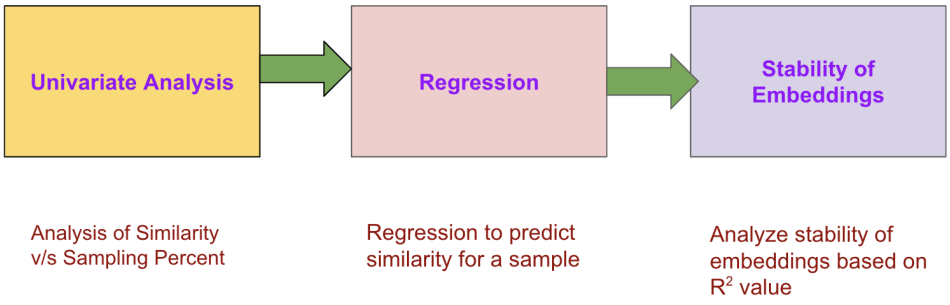


Figure 4.1: Graphical Representation of our contributions. It shows the step-by-step process followed to answer questions from Chapter 1.

4.1 Univariate Analysis

As discussed in previous chapter, we have sampling size, graph properties, and Euclidean distances. As the distance has no bounds, we first bound the distance between zero and one using the formula as per Y. Bai et al. [1]:

$$\text{Similarity} = e^{-y}, \tag{4.1}$$

where y is Euclidean distance. The equation (4.1) converts Euclidean distance to similarity.

First, we try to analyze how the similarity changes with respect to sampling size by gathering 15 samples for each graph and for each sampling size. There are ten graphs and eight sampling size and 15 runs. Thus, in total we have 1,200 instances. Next, instances are grouped based on sampling size and we take the median of them. The points on the graph are these medians. The reason the average is not considered is because it is sensitive to the outliers in the dataset. This analysis is performed for each embedding and sampling method. Figures 4.2 to 4.5 are for graphs from diverse domains. In these figures, we observe

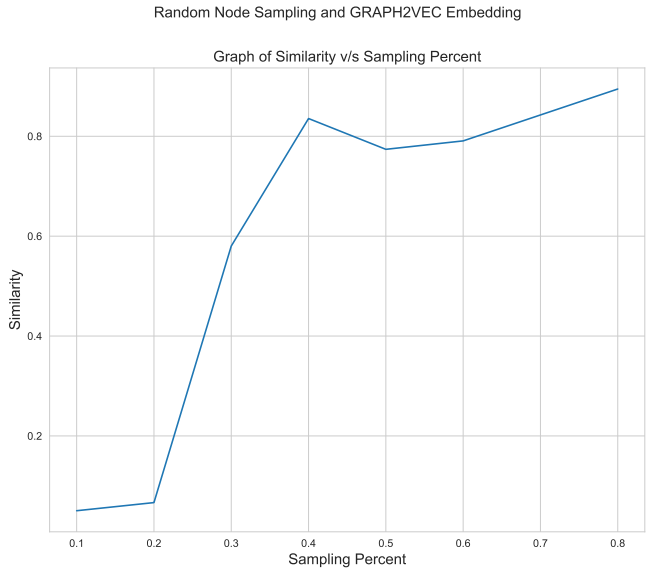


Figure 4.2: Similarity versus Sampling Size for Random Node Sampling and Graph2vec embedding. This gives intuitively minimum 40% sample required for 80% similarity with original graph

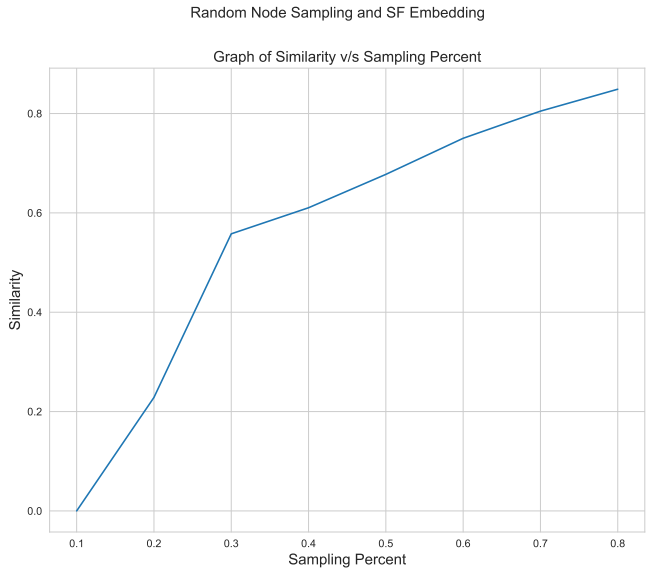


Figure 4.3: Similarity versus Sampling Size for Random Node Sampling and Spectral Features embedding. This gives intuitively minimum 30% sample required for nearly 60% similarity with original graph

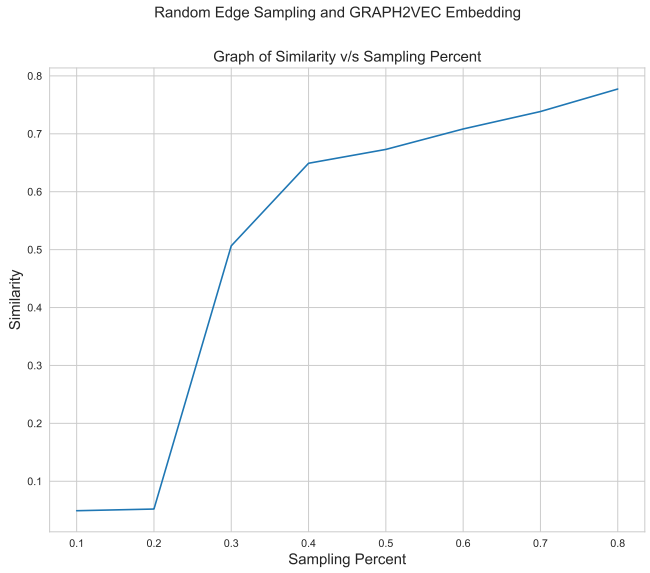


Figure 4.4: Similarity versus Sampling Size for Random Edge Sampling and Graph2Vec Embedding. This gives intuitively minimum 40% sample required for 65% similarity with the original graph

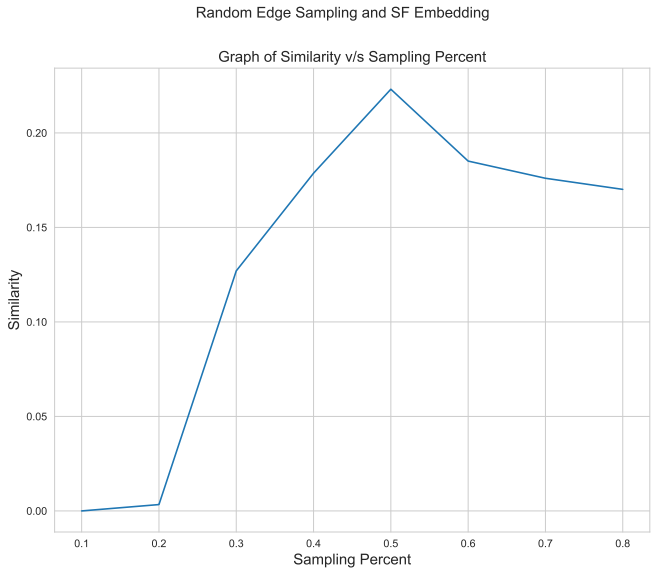


Figure 4.5: Similarity versus Sampling Size for Random Edge Sampling and Spectral Features embedding. This gives intuitively minimum 50% sample required for 25% similarity with original graph

In **Figure 4.2**, the similarity rapidly increases with sampling size from 0.2 to 0.4, after which it remains nearly stable. Thus at 40% sample, we get close to 83% similarity to the original graph. In other words, if we analyze a 40% sample, our embeddings are 83% similar compared to that of the original graph.

Figure 4.3 shows similarity versus sampling curve for Random Node Sampling and Spectral Features (SF) embedding. As we can see, 50% sampling gives up to 70% similarity. It also suggests that sampling size cannot be less than 30%. In other words, any value above 30% will work as the similarity keeps increasing with sampling size after 30%.

Figure 4.4, shows similarity versus sampling for Random Edge Sampling and Graph2Vec embedding. Here too, close to 50-55% gives 70% similarity to the original graph. Thus, the graph with half the size of the original is enough to analyze and would be nearly as good as the original graph.

Figure 4.5 shows similarity versus sampling for Random Edge Sampling and Spectral Features embedding. The 2D plot is similar to others, but it gives the maximum similarity of 22-24% for 50% sampling. The similarity is below 70%. As the similarity is not that stable as after certain sampling size (50%), the similarity starts to drop.

The goal of this analysis is to obtain an intuition of how much a small the sample can be. However, this just considers the sample size and similarity. The way we calculated the similarity was through embeddings. All the embeddings capture the connectivity of the graph differently. So it is incorrect to conclude anything just based on single variable analysis. So, we decided to consider various properties of the graphs that capture connectivity from different angles.

The properties that we will be considering are discussed in the section 3.3 in chapter 3. There are a total of 21 properties. Now question comes up such as *how do we check these properties? Can we check individually?* or *can we assess the overall effect of all these properties?* If we check individually, it will be same as our earlier analysis. Therefore, it will not be sufficient to conclude based on individual properties. Thus, we drop this option. Next, we try to figure out the overall effect of all these properties. One way is to calculate the graph properties for multiple instances

and then take a weighted average of all of these properties. Average will be influenced by majority values and might give incorrect results. Another way is to calculate the median. Median will work for one property, but calculating across various properties will mislead the results. The primary reason being the scale for all properties being different. Scaling hints us that properties need to be normalized to bring them to the same scale. Scaling every feature can be done for a limited number of features. However, data will keep increasing, and hence the number of features. Thus, manual work will have limitations for the huge amount of data and a large number of features. Additionally, manual work will be also limited to the number of datasets at present. What if we need determine the results for a completely new dataset? This takes us on the route of using a machine learning model. We need to train model on existing data or graphs for model to learn the properties of graphs. The model developed by providing training data for learning is called as a supervised learning model. Supervised machine learning models are mainly of two types: Classification and Regression. Let us understand them briefly, so we know why we selected one method over the other.

Classification is used when we are predicting discrete values, e.g., Classifying credit card transactions as legitimate or not. This is saying a yes or no. This "yes" or "no" are called as labels. When there are two labels, it is binary classification, else it is multi-label classification.

Regression, on the other hand, is used when we are predicting continuous values, e.g., given weather and rainfall data of one year, what is the amount of rain we will receive today? Thus, here we predict actual value rather than a yes or no. In our case, we are predicting similarity which is a continuous value. Hence, we can formulate our problem as a regression problem. Additionally, as we were trying to calculate average across all properties, earlier, the regression will also do the same but in an optimized manner based on the kind of model. Hence, regression solves the challenge of finding averages and predicting on new datasets.

The question we are asking is, *given a sampling size, and the properties for that sampling size of the graph, how much similarity we can get?* Next, let us see how the regression is performed and how different variables affect the output.

4.2 Regression for Similarity Prediction

We performed the set of experiments in two sets. The reason being if domain plays a role in prediction or not. So, we first train the regression model with only social media datasets and then add more datasets.

4.2.1 Experiments on Social Media Networks

Dataset descriptions are given in chapter 3. From all the datasets mentioned, we first consider (a) Facebook GEMSEC dataset and (b) Twitch Social Network datasets. We consider the basic properties defined by SNAP website [12] for those datasets for these two datasets. Then we follow as outlined next:

Sampling from Datasets and Computing Graph Properties

As discussed in Chapter 3, we first collected the data for the facebook and twitch datasets. The collected data included sampling sizes (10-80%) and graph properties respective to those samples. The graph properties we considered are: sampling size, clustering coefficient, average shortest path length, number of edges, number of nodes, diameter, transitivity, and density. These properties along with the sample size will be our features for the model.

Shuffling Datasets

Next, we shuffle the dataset to prevent models from learning the order. In simple words, we want to make sure model treats each data point or row as independent and that it is unrelated to the points above or below it in the dataset.

Scaling the values

As all the features have different scale, we need to bring them all to the same scale. This process is called as standardization. Standardization means to subtract the mean from every instance

and divide by the standard deviation. The formulation is as follows:

$$z = (x - u)/s \tag{4.2}$$

where u is the mean of the training samples and s is the standard deviation of the training samples.

The reason to do standardization of the data is to prevent model from being biased on scale thereby resulting in incorrect predictions. We use StandardScaler from package sklearn [16] and API [2]. The dependent variable is Euclidean distance and to convert it to a value between zero and one, we use e^{-y}, as specified by Bai et.al [1] where y is Euclidean distance.

Test/Train Split

We have the dataset preprocessed and ready to analyze. We will divide our dataset into train and test sets. The reason is to train the model for learning the pattern in the data and the test for some arbitrary data to verify model's performance. In the regression, we check the performance by amount of error in the actual versus predicted values. So, we divide the dataset into 70% train and 30% test. Meaning that 70% of the data will be shown to model and 30% will be hidden. For the splitting, we use test_train_split method from Sklearn [2].This method also guarantees that the distribution of data is even in the train and the test sets. We finally perform linear regression by fitting the model on the train dataset and predicting on the test dataset. We calculate Root Mean Square Error (RMSE) to measure the performance of model. We note down the RMSE for different embedding methods and sampling approaches as shown in the Table 4.1. As we can see, Tables 4.2 and 4.1 have similar values, indicating that the model is not overfitted.

Perform Ordinary Least Squares Regression

We get the RMSE values from the regression model, but how do we know the RMSE we got is good or not, and which properties are significant or contributed most to model? To answer these questions, we analyze using OLS (Ordinary Least Square) method from statsmodel package

Sampling Method	Embedding Method	
	Graph2Vec	**Spectral Features (SF)**
Random Node Sampling	0.17	0.18
Random Edge Sampling	0.17	0.13

Table 4.1: RMSE of Test Dataset for Graph2Vec and Spectral Features embeddings calculated for Random Node and Random Edge Sampling methods. The dataset contains social networks.

Sampling Method	Embedding Method	
	Graph2Vec	**Spectral Features (SF)**
Random Node Sampling	0.17	0.17
Random Edge Sampling	0.16	0.14

Table 4.2: RMSE of Train Dataset for Graph2Vec and Spectral Features embeddings calculated for Random Node and Random Edge Sampling methods. The dataset contains social networks.

[25]. The OLS is another name for linear regression. The interesting thing about the statsmodel package is that it gives a detailed summary of the model including some statistical measures. The summary table for Random node Sampling and Spectral Features embedding is presented in Figure 4.6

Let us understand some important terms that we are mainly considering. We have a total of eight independent variables or features denoted by $x1$ to $x8$. R-squared captures the size of variation in the dependent variable that is explained by the independent variables. Here, 72.3% variation in distances is explained by $x1$ to $x8$. The drawback of R-squared is that it is directly proportional to the input or predictors or output variable that makes the value inconclusive in

some cases. Hence, the next term *Adjusted R-squared*, which is the improved version of R-squared that is adjusted for the number of variables. F-statistic shows the overall significance of regression with multiple features. It is calculated as:

$$F\text{-statistic} = \frac{\text{the mean regression sum of squares}}{\text{the mean error sum of squares}} \tag{4.3}$$

F-statistic values can range from zero to an arbitrary large number. However, this alone does not determine significance. There is one more value Prob(F-Statistic). It evaluates the significance level of all the variables together unlike the t-statistic that evaluates it for individual variables. It depicts the probability of the null hypothesis being true. As per the results in Table 4.3, the probability is zero. This implies that the overall regressions is meaningful. For all independent variables, we have ($P > |t|$) values. Here, P is denotes the p-value and tests the null hypothesis that the variable has no correlation with the dependent variable. That means, change in one input variable has no effect on the output variable. To decide when to reject null hypothesis, we decide significance level or threshold before experiment. In general, the value of this threshold is 0.05. So, if the p-value of input variables is less than threshold, the data least agrees with hypothesis. Thus, we reject null hypothesis that there is no correlation between input and output variables. To map this in our case, if any of the variables ($x1, ..., x8$) has p-value$<$ 0.05, then the change in that variable will affect the similarity significantly.

Table 4.3 summarizes all values at once. A point to note is that Adjusted R-Squared from OLS summary is very similar to *10-fold cross-validation* calculated using cross_val_score function from sklearn library [16].

Sampling Method	**Embedding Method**	F-**Statistic**	**Adjusted** R-**Squared**
Random Node Sampling	Graph2Vec	1509.0	0.72
	Spectral Features	2876.0	0.76
Random Edge Sampling	Graph2Vec	52.21	0.70
	Spectral Features	15.69	0.18

Table 4.3: Summary of F-Statistic and R-Squared for Social Network Datasets. It shows OLS Summary for all embedding and sampling methods for all significant features of regression model.

OLS Regression Results

Dep. Variable:	distances	**R-squared:**	0.723
Model:	OLS	**Adj. R-squared:**	0.722
Method:	Least Squares	**F-statistic:**	1509.
Date:	Sun, 18 Apr 2021	**Prob (F-statistic):**	0.00
Time:	15:02:55	**Log-Likelihood:**	1485.3
No. Observations:	4642	**AIC:**	-2953.
Df Residuals:	4633	**BIC:**	-2895.
Df Model:	8		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	0.4668	0.003	180.782	0.000	0.462	0.472
x1	0.2798	0.006	43.471	0.000	0.267	0.292
x2	0.1064	0.005	20.268	0.000	0.096	0.117
x3	-0.0962	0.007	-13.754	0.000	-0.110	-0.082
x4	-0.1433	0.006	-22.416	0.000	-0.156	-0.131
x5	0.1317	0.006	20.320	0.000	0.119	0.144
x6	-0.1506	0.006	-25.919	0.000	-0.162	-0.139
x7	0.0685	0.007	9.793	0.000	0.055	0.082
x8	-0.0123	0.004	-3.063	0.002	-0.020	-0.004

Omnibus:	231.886	**Durbin-Watson:**	1.967
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	186.012
Skew:	0.408	**Prob(JB):**	4.06e-41
Kurtosis:	2.455	**Cond. No.**	6.95

Figure 4.6: Ordinary Least Square (OLS) Summary: Random Node Sampling and Spectral Features Embedding generated by statsmodel package [25]

We follow the same process for other embedding and sampling methods.

OLS Regression Results

Dep. Variable:	distances	**R-squared:**	0.760
Model:	OLS	**Adj. R-squared:**	0.760
Method:	Least Squares	**F-statistic:**	2876.
Date:	Sun, 18 Apr 2021	**Prob (F-statistic):**	0.00
Time:	14:56:40	**Log-Likelihood:**	2574.3
No. Observations:	7255	**AIC:**	-5131.
Df Residuals:	7246	**BIC:**	-5069.
Df Model:	8		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	0.4737	0.002	237.552	0.000	0.470	0.478
x1	0.3375	0.005	68.933	0.000	0.328	0.347
x2	0.0705	0.004	18.303	0.000	0.063	0.078
x3	-0.0739	0.005	-14.475	0.000	-0.084	-0.064
x4	-0.0710	0.005	-14.576	0.000	-0.081	-0.061
x5	0.0472	0.005	9.918	0.000	0.038	0.057
x6	-0.1410	0.004	-31.871	0.000	-0.150	-0.132
x7	-0.0322	0.005	-6.028	0.000	-0.043	-0.022
x8	-0.0154	0.003	-5.630	0.000	-0.021	-0.010

Omnibus:	406.303	**Durbin-Watson:**	2.042
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	475.926
Skew:	0.619	**Prob(JB):**	4.51e-104
Kurtosis:	3.208	**Cond. No.**	6.66

Figure 4.7: Ordinary Least Square (OLS) Summary: Random Node Sampling and Graph2Vec Embedding generated by statsmodel package [25]

OLS Regression Results

Dep. Variable:	distances	**R-squared:**	0.715
Model:	OLS	**Adj. R-squared:**	0.701
Method:	Least Squares	**F-statistic:**	52.21
Date:	Thu, 22 Apr 2021	**Prob (F-statistic):**	9.38e-32
Time:	11:10:19	**Log-Likelihood:**	47.419
No. Observations:	132	**AIC:**	-80.84
Df Residuals:	125	**BIC:**	-60.66
Df Model:	6		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	0.5396	0.015	34.918	0.000	0.509	0.570
x1	0.1750	0.029	5.939	0.000	0.117	0.233
x2	-0.0400	0.020	-1.991	0.049	-0.080	-0.000
x3	-0.1661	0.041	-4.052	0.000	-0.247	-0.085
x4	-0.1702	0.070	-2.442	0.016	-0.308	-0.032
x5	0.2255	0.044	5.091	0.000	0.138	0.313
x6	0.2316	0.065	3.556	0.001	0.103	0.361

Omnibus:	12.705	**Durbin-Watson:**	1.767
Prob(Omnibus):	0.002	**Jarque-Bera (JB):**	17.311
Skew:	0.534	**Prob(JB):**	0.000174
Kurtosis:	4.417	**Cond. No.**	10.9

Figure 4.8: Ordinary Least Square(OLS) Summary: Random Edge Sampling and Graph2Vec Embedding generated by statsmodel package [25]

OLS Regression Results

Dep. Variable:	distances	**R-squared:**	0.195
Model:	OLS	**Adj. R-squared:**	0.183
Method:	Least Squares	**F-statistic:**	15.69
Date:	Sat, 24 Apr 2021	**Prob (F-statistic):**	1.65e-11
Time:	20:55:42	**Log-Likelihood:**	135.95
No. Observations:	264	**AIC:**	-261.9
Df Residuals:	259	**BIC:**	-244.0
Df Model:	4		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	0.1524	0.009	16.899	0.000	0.135	0.170
x1	0.0201	0.010	1.937	0.054	-0.000	0.041
x2	0.0991	0.027	3.685	0.000	0.046	0.152
x3	0.0455	0.010	4.513	0.000	0.026	0.065
x4	-0.0717	0.027	-2.695	0.007	-0.124	-0.019

Omnibus:	39.165	**Durbin-Watson:**	2.056
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	51.854
Skew:	1.049	**Prob(JB):**	5.50e-12
Kurtosis:	3.561	**Cond. No.**	6.60

Figure 4.9: Ordinary Least Square(OLS) Summary: Random Edge Sampling and Spectral Features Embedding generated by statsmodel package [25]. This shows summary for significant features.

Here, as we can see, there are just four important features $(x1, x2, x3,$ and $x4)$. The reason being other features had higher p-values than our threshold of 0.05, which means those features do not contribute significantly to predicting similarity. Another perspective to think about features

is, we can still keep the features we removed as they contribute but only slightly. In case of graphs, the more structural properties we gather, the more information we capture. This is also captured by the increased R-Squared value. Figure 4.10 shows the summary for Random Edge Sampling and Spectral Features Embedding with all input variables available. As we can see, for all features, R-Squared increased slightly but F-Statistic decreased. This decrease in F-Statistics shows model is less significant.

OLS Regression Results

Dep. Variable:	distances	**R-squared:**	0.226
Model:	OLS	**Adj. R-squared:**	0.202
Method:	Least Squares	**F-statistic:**	9.318
Date:	Sat, 24 Apr 2021	**Prob (F-statistic):**	2.87e-11
Time:	21:15:59	**Log-Likelihood:**	147.95
No. Observations:	264	**AIC:**	-277.9
Df Residuals:	255	**BIC:**	-245.7
Df Model:	8		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	0.1523	0.009	17.501	0.000	0.135	0.169
x1	0.0552	0.018	2.990	0.003	0.019	0.092
x2	0.0658	0.029	2.236	0.026	0.008	0.124
x3	0.0610	0.022	2.825	0.005	0.018	0.104
x4	-0.0405	0.043	-0.951	0.342	-0.124	0.043
x5	-0.0345	0.020	-1.688	0.093	-0.075	0.006
x6	-0.0587	0.032	-1.831	0.068	-0.122	0.004
x7	-0.0163	0.040	-0.403	0.687	-0.096	0.063
x8	-0.0267	0.013	-2.137	0.034	-0.051	-0.002

Omnibus:	35.998	**Durbin-Watson:**	2.199
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	47.182
Skew:	1.028	**Prob(JB):**	5.68e-11
Kurtosis:	3.257	**Cond. No.**	12.4

Figure 4.10: OLS Summary: Random Edge Sampling and Spectral Features Embedding all features generated by statsmodel package [25].This shows summary for all features.

Next, we also try LASSO and Ridge regression, but LASSO gives RMSE more or less equal to Linear Regression. Ridge regression too was nearly close to value. Hence, we consider Linear Regression model for further analysis for clarity.

Performance on Validation Datasets

We checked how our model performs on test data. The goal is to see how generalized the model is by validating it on different datasets. We performed validation on the social media dataset and a dataset from collaboration network. Table 4.4 shows actual versus predicted values for social media dataset for each embedding and sampling method.

Validation Datasets that we used are: "Astro Physics collaboration network" and "Social Network from Facebook page to page network for TV Shows category". To review datasets please refer Chapter 3. We took one sample for every sampling size (percentage) for both the datasets.

Sampling Methods	**TV Show Page to Page Network**		**Astrophysics Network**	
	Graph2Vec	**Spectral Features (SF)**	**Graph2Vec**	**Spectral Features (SF)**
Random Node Sampling	0.29	0.17	0.23	0.29
Random Edge Sampling	0.17	0.12	0.27	0.11

Table 4.4: RMSE for Facebook TV Show Page to Page Network and Astrophysics Network. These networks are used to validate how the model performs on new datasets

From the Table 4.4, we can see Graph2Vec embedding for Random Node Sampling and Spectral Features (SF) embedding for Random Edge Sampling gives nearly the same RMSE for both datasets. However, SF embedding for Random Node and Graph2Vec embedding for Random Edge shows difference of RMSE. One reason for such difference in values is also because all the four combinations do not have the exact same number of instances. The reason being both embedding methods took very long time to embed graphs after random edge sampling. Another reason is also because Astrophysics is a large dataset with 18k nodes and 198k edges. So although these numbers look close and good, but they are not telling the complete story. To understand the full story, let us visualize them.

Let us see the actual versus predicted plots for both the validation datasets. X-axis shows the sampling size. Y-axis shows Normalized Euclidean Distance, i.e., similarity. Next, we discuss insights from Figures 4.11 to 4.18. As we can see in all figures, the plot for TV Show Facebook Page-to-Page Network is better than astrophysics network.

For TV Show Dataset:

For Random Node Sampling with Graph2Vec and SF Embedding (Figure 4.11), the plots are nearly perfect, i.e., model is able to predict all the values with minimal errors. The reason being validation dataset belongs to the same domain of train dataset, i.e., social networks. For Edge Sampling though, graph2vec embedding is again good as expected but the plot is completely opposite for SF embedding, where a single point somehow matches but for the rest of the data points, difference between actual and predicted is large.

For Astrophysics Dataset:

For this dataset, there is a vast difference between actual and the predicted values of similarity. For Random Edge sampling, the curves do not even overlap in Figures 4.16 and 4.18. What does this show? Our model works the best for dataset for Social Media domain but cannot be applied for other domains. That makes sense because properties of every network are different, e.g., degree distribution in social networks follow power law distribution but it is not the same case for, e.g., collaboration networks. What does this tell us? For a model to be generalized, it should understand how one property behaves differently when the network changes. Thus, we decide to add datasets from other domains as well as various properties.

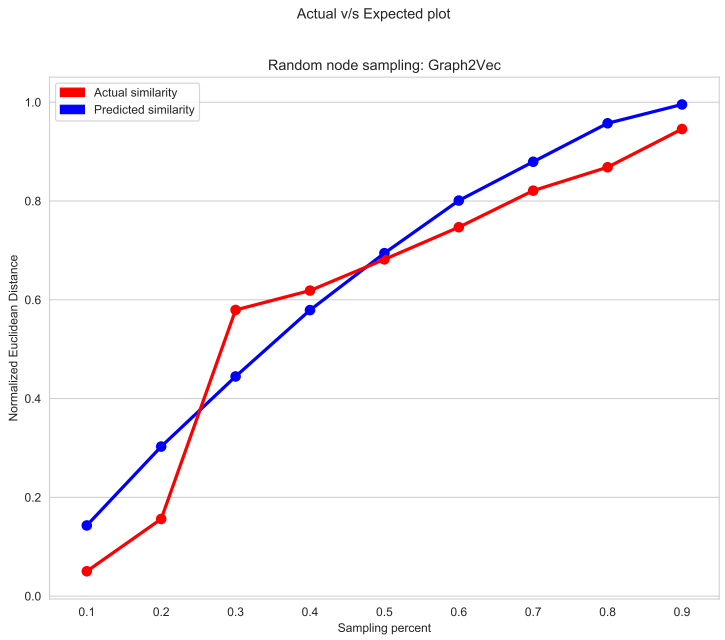


Figure 4.11: Actual versus Predicted Similarity for TV Show Validation Dataset: Random Node and Graph2vec embedding

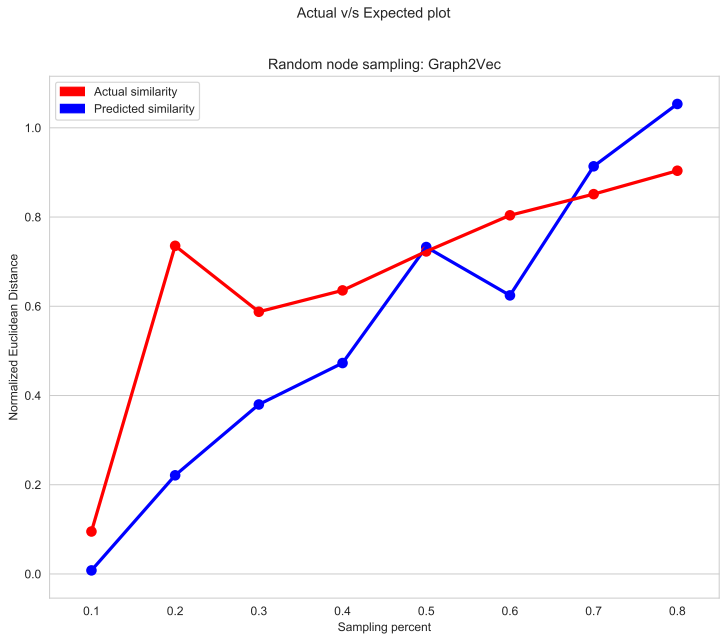


Figure 4.12: Actual versus Predicted Similarity for Astrophysics Dataset: Random Node Sampling and Graph2vec embedding

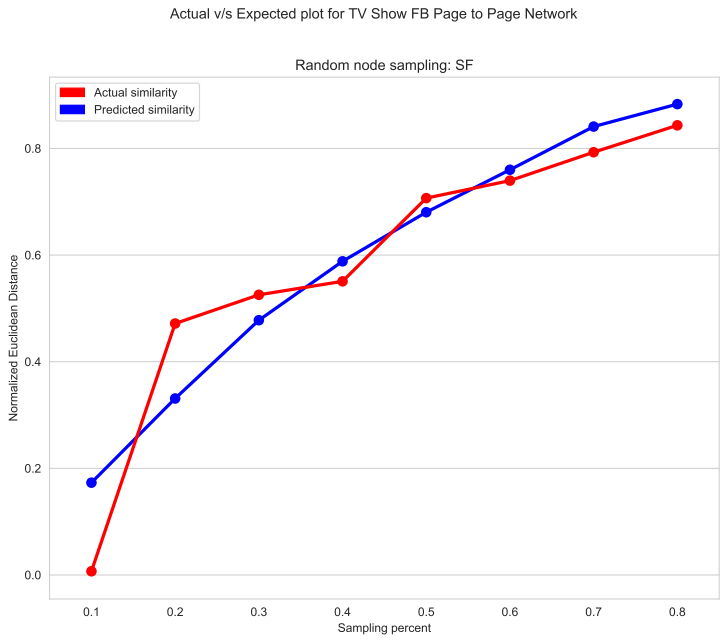


Figure 4.13: Actual versus Predicted Similarity for TV Show Validation Dataset: Random Node and SF embedding

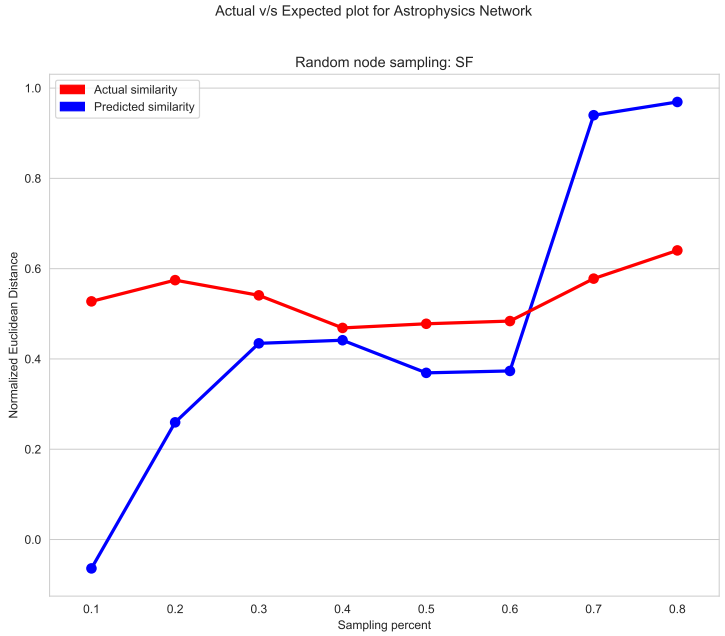


Figure 4.14: Actual versus Predicted Similarity for Astrophysics Dataset: Random Node and SF embedding

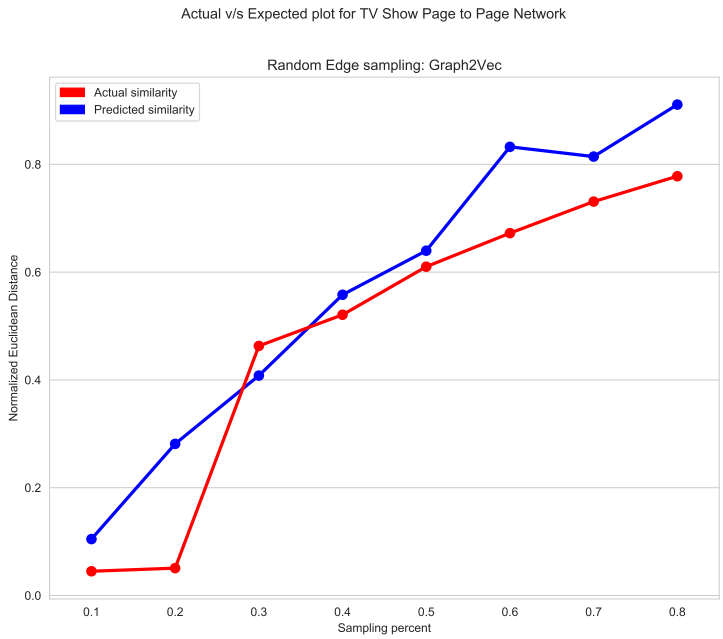


Figure 4.15: Actual versus Predicted Similarity for TV Show Validation Dataset: Random Edge and Graph2Vec embedding

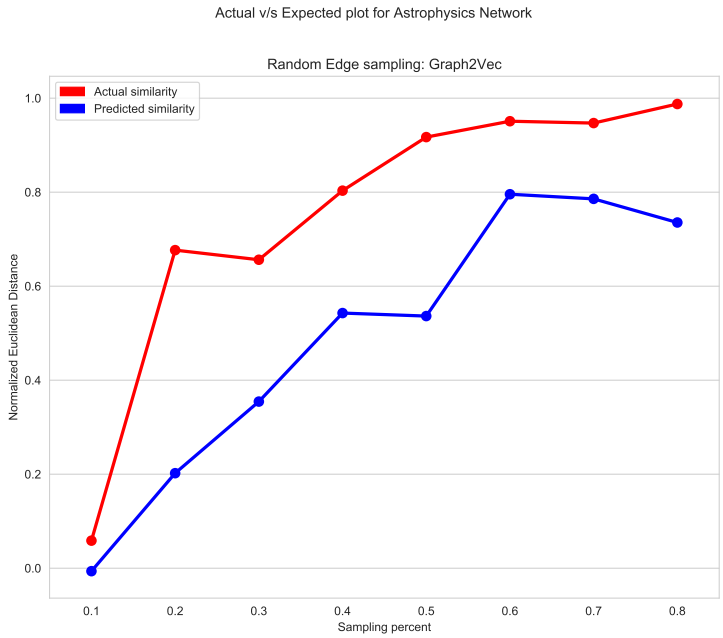


Figure 4.16: Actual versus Predicted for Astrophysics Dataset: Random Edge and Graph2Vec embedding

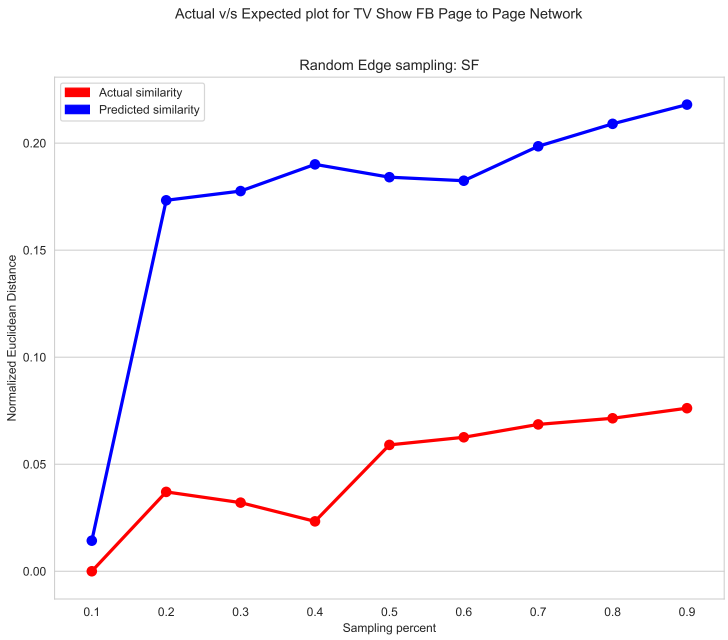


Figure 4.17: Actual versus Predicted Similarity for TV Show Validation Dataset: Random Edge and SF embedding

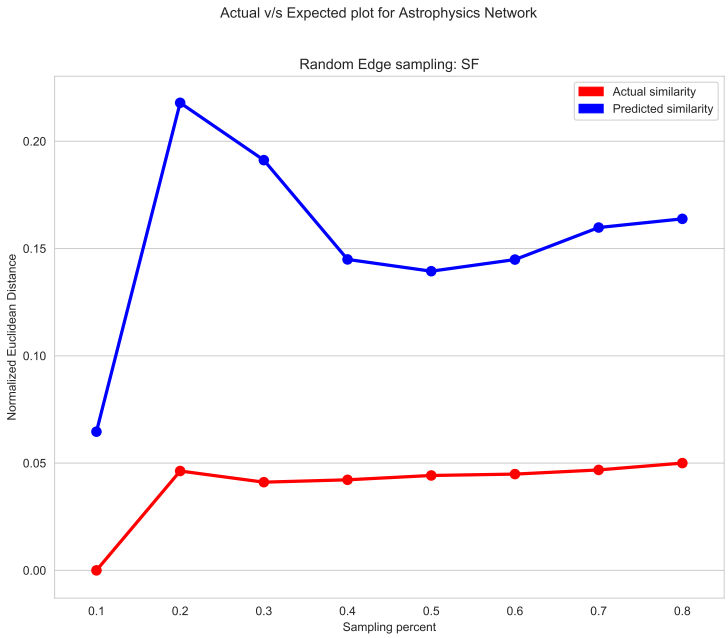


Figure 4.18: Actual versus Predicted Similarity for Astrophysics Dataset: Random Edge and SF embedding

4.2.2 Experiment with Multi-domain Networks

As we saw in previous figures, our model did not do a good job in predicting the similarities for graphs from domains other than social media. So, we decide to add datasets from other domains. Additionally, graph properties also behaves differently depending on domain. We decided to add datasets incrementally instead of adding them all at once. Hence, we add datasets from collaboration, peer-to-peer, and autonomous systems domain. We also add properties which capture the connectivity of graphs. We try to include properties which capture the underlying structure of graph as well as those that are faster to compute. The point to note here is that we only capture properties which are calculated at the Network/Graph level and not at the node level. For example, centrality is calculated at the node level so we do not consider it. The graph properties we consider are: clustering coefficient, average shortest path length, edges, nodes, diameter, transitivity, density, assortativity coefficient, Estrada index, circuit Rank, girth, wiener index, independence number, annihilation number, average degree, second moment, third moment, fourth moment, radius, and residue. Next, we will deep dive into the experiment following a step by step process.

Multiple Samples

We now consider all datasets outlined in our dataset section (page 13) and all graph properties (described on page 29). Facebook dataset contains many Page-to-Page networks which increases number of graphs for social media domain. Hence, we remove two network graphs: company and athletes from it to make the dataset balanced. Next, we sample the graph between 10% and 80% and take 15 samples for each sample size. So, we have $15 \times 8 = 120$ samples for each dataset. There are ten types of graphs from different domains and 15 samples or subgraphs per domain yielding 150 instances per sample. we calculate the graph properties mentioned in section 3.3 for all these subgraphs.

Shuffle and Preprocessing dataset

Next, we shuffle datasets five times to avoid the model from learning order dependency. Now, we need to preprocess the dataset before giving it to the model. Just like we did in previous experiment, we will scale the independent variables or features using Standard Scaler. For output or dependent variable, we need the value to be between zero and one. So for every value, we will calculate e^{-y}, where y is dependent variable or output which is similarity in our case.

Test-Train Split

Now, we have a cross-domain (multi-domain) dataset of subgraphs, sampling sizes, respective graph properties, and similarity. We feed this data to the regression model to predict similarity. First, we divide the dataset into 70% train and 30% test. Next, we perform Linear Regression on the dataset by training with train data and predicting on test data. We create regression model for every pair of Random Node Sampling, Random Edge Sampling with Graph2Vec and Spectral Features Embedding. We again create table of RMSE for test dataset. To remind, table 4.1 shows RMSE for previous experiments for social media networks.

Sampling Method	**Embedding Method**	
	Graph2Vec	**Spectral Features (SF)**
Random Node Sampling	0.14	0.14
Random Edge Sampling	0.12	0.11

Table 4.5: RMSE for regression model on Test Dataset. The dataset contains graphs from social, collaboration, autonomous systems, and peer to peer networks i.e., multi domain networks

All the four values in the dataset are nearly same. So it is difficult to get insights on these values. We run Ordinary Least Squares (OLS) from stats models package [25]. The summary

created by statsmodels package will give more insights from model. The figures shows summary given by OLS model for each embedding and sampling method.

(a) **Random Node Sampling and Graph2Vec Embedding**:

Figure 4.19 shows the OLS summary for Random Node Sampling and Graph2Vec Embedding. It contains all the features described in previous chapter except first moment as it is always zero and matching number. We also remove edges as a feature as the summary gives multicollinearity error. With all the features, our Adjusted R-Squared is 0.83 which means our model is able to predict correctly 83% of the time. We see if all the features contribute significantly. So, we will check $P > |t|$ column which will give p-value for each feature. Our threshold is again 0.05 to reject the null hypothesis. So, we will take features which has this p-value less than 0.05 (our threshold). Thus, we consider features which are contributing significantly to predict our output. Finally, as shown in Figure 4.20, all 10 features are significant. However, as we can see, Adjusted R-Squared is decreased to 0.81 and F-statistic increased from 203 to 353 which means the current set of features are more significant. We removed girth as it needs more computation time for large graphs and one of our validation datasets is large. Moreover, adding or removing girth does not change model significantly. Feature $x7$ is the girth in Figure 4.20. Thus, we can safely remove girth from our feature set. So, we can say that if we have the 9 features, and sampling ratio is between 10%-80%, our model predicts 81% more accurately. These 9 features are: sample, transitivity, nodes, density, assortativity coefficient, circuit Rank, Wiener index, independence number, and third moment.

(b) **Random Node Sampling and SF Embedding**

The same experiment is performed for SF embedding and in the same setting, i.e., same instances and datasets. Figures 4.21 and 4.22 show the OLS summary for the model with all features and significant features. Here, we again removed 'girth' and 'edges' to solve multicollinearity problem. Girth seems to be capturing the same properties as that of transitivity. In other words, transitivity and girth has some correlation. Also, if we observe, sample($x1$) p-value

is greater than our threshold. It means that sample is not contributing significantly to the similarity prediction. We will still keep the sampling size as our graph properties are for that sampling size and coefficient is not too small. The Adjusted R-Squared here is 0.78, which means 78% of the time it predicts the data correctly. The features we select here are: Sample, Clustering Coefficient, Average Shortest Path Length, Transitivity, Circuit Rank, Wiener Index, Independence Number, Annihilation Number, Second Moment, and Fourth Moment.

(c) **Random Edge Sampling and Graph2Vec Embedding**:

Figures 4.23 and 4.24 show OLS summary for Random Edge sampling and Graph2Vec embedding. Initially, we considered all the features and ran the OLS on that. On checking summary, we got the error that input variables have correlation. On analyzing, we removed girth and nodes from features and the error was fixed. Figures 4.23 and 4.24 show summaries after the error was fixed. As we did before, we selected features with p-value less than 0.05 and reduce the number of features from 19 to 11. Adjusted R-Squared is not changed much but there is increase in F-statistic which gives significance of complete model. So we can say our model which has 11 features is more significant. As per adjusted R-Squared, our model can predict the similarity correctly 86% of the time. The significant features are: Sample, Average Shortest Path Length, Edges, Density, Circuit Rank, Wiener Index, Annihilation Number, Average Degree, Third Moment, Fourth Moment, and Radius

(d) **Random Edge Sampling and SF Embedding**: Figures 4.25 and 4.26 show OLS summaries for Random edge sampling and SF embedding. Compared to other models, Adjusted R-Squared here is less along with F-statistic. Here again we take significant features by considering p-Model can accurately predict correctly 57% of the time. The significant features are: Sample, Average shortest path length, edges, transitivity, density, circuit Rank, Independence Number, Annihilation number, Average degree, Second Moment, Third Moment, and Residue.

OLS Regression Results

Dep. Variable:	distances	**R-squared:**	0.835
Model:	OLS	**Adj. R-squared:**	0.831
Method:	Least Squares	**F-statistic:**	203.3
Date:	Tue, 27 Apr 2021	**Prob (F-statistic):**	5.40e-298
Time:	21:47:59	**Log-Likelihood:**	428.20
No. Observations:	825	**AIC:**	-814.4
Df Residuals:	804	**BIC:**	-715.4
Df Model:	20		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	0.5822	0.005	114.251	0.000	0.572	0.592
x1	0.1916	0.020	9.619	0.000	0.152	0.231
x2	-0.0079	0.016	-0.485	0.628	-0.040	0.024
x3	-0.0020	0.017	-0.117	0.907	-0.036	0.032
x4	-0.0578	0.035	-1.639	0.102	-0.127	0.011
x5	-0.1080	0.028	-3.914	0.000	-0.162	-0.054
x6	1.4681	0.133	11.057	0.000	1.207	1.729
x7	0.0406	0.010	4.230	0.000	0.022	0.060
x8	0.0509	0.016	3.122	0.002	0.019	0.083
x9	-0.0032	0.005	-0.651	0.515	-0.013	0.006
x10	-0.1543	0.032	-4.754	0.000	-0.218	-0.091
x11	-0.0154	0.006	-2.578	0.010	-0.027	-0.004
x12	-0.3007	0.034	-8.745	0.000	-0.368	-0.233
x13	-1.1467	0.091	-12.579	0.000	-1.326	-0.968
x14	-0.0874	0.098	-0.897	0.370	-0.279	0.104
x15	-0.0148	0.042	-0.354	0.723	-0.097	0.067
x16	-0.0210	0.028	-0.746	0.456	-0.076	0.034
x17	-0.0467	0.008	-6.182	0.000	-0.062	-0.032
x18	-0.0213	0.025	-0.859	0.391	-0.070	0.027
x19	0.0314	0.037	0.856	0.393	-0.041	0.104
x20	0.1824	0.067	2.719	0.007	0.051	0.314

Figure 4.19: Ordinary Least Squares (OLS) Summary: Random Node Sampling and Graph2Vec Embedding for all features

OLS Regression Results

Dep. Variable:	distances	**R-squared:**	0.813
Model:	OLS	**Adj. R-squared:**	0.811
Method:	Least Squares	**F-statistic:**	353.6
Date:	Wed, 28 Apr 2021	**Prob (F-statistic):**	2.85e-288
Time:	15:39:14	**Log-Likelihood:**	384.60
No. Observations:	825	**AIC:**	-747.2
Df Residuals:	814	**BIC:**	-695.3
Df Model:	10		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	0.5905	0.005	110.805	0.000	0.580	0.601
x1	0.1908	0.019	10.139	0.000	0.154	0.228
x2	-0.1280	0.016	-8.057	0.000	-0.159	-0.097
x3	1.4001	0.076	18.428	0.000	1.251	1.549
x4	0.0433	0.008	5.334	0.000	0.027	0.059
x5	0.0453	0.013	3.484	0.001	0.020	0.071
x6	-0.1897	0.014	-13.847	0.000	-0.217	-0.163
x7	-0.0146	0.006	-2.300	0.022	-0.027	-0.002
x8	-0.3338	0.021	-15.699	0.000	-0.376	-0.292
x9	-0.9022	0.059	-15.346	0.000	-1.018	-0.787
x10	-0.0308	0.008	-4.046	0.000	-0.046	-0.016

Omnibus:	51.305	**Durbin-Watson:**	2.012
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	59.328
Skew:	0.644	**Prob(JB):**	1.31e-13
Kurtosis:	3.260	**Cond. No.**	37.7

Figure 4.20: Ordinary Least Squares (OLS) Summary: Random Node Sampling and Graph2Vec Embedding for Significant Features

OLS Regression Results

Dep. Variable:	distances	**R-squared:**	0.790
Model:	OLS	**Adj. R-squared:**	0.785
Method:	Least Squares	**F-statistic:**	164.6
Date:	Wed, 28 Apr 2021	**Prob (F-statistic):**	1.43e-266
Time:	21:25:43	**Log-Likelihood:**	429.31
No. Observations:	854	**AIC:**	-818.6
Df Residuals:	834	**BIC:**	-723.6
Df Model:	19		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	0.5487	0.005	107.990	0.000	0.539	0.559
x1	0.0322	0.020	1.640	0.101	-0.006	0.071
x2	0.1065	0.016	6.609	0.000	0.075	0.138
x3	-0.0366	0.018	-2.062	0.039	-0.071	-0.002
x4	0.0788	0.033	2.355	0.019	0.013	0.144
x5	-0.0661	0.028	-2.383	0.017	-0.120	-0.012
x6	-0.0596	0.128	-0.465	0.642	-0.311	0.192
x7	-0.0137	0.010	-1.416	0.157	-0.033	0.005
x8	0.0282	0.017	1.676	0.094	-0.005	0.061
x9	0.0070	0.006	1.084	0.279	-0.006	0.020
x10	-0.1375	0.032	-4.339	0.000	-0.200	-0.075
x11	-0.3125	0.034	-9.163	0.000	-0.379	-0.246
x12	-0.2865	0.095	-3.008	0.003	-0.473	-0.100
x13	1.1233	0.102	10.985	0.000	0.923	1.324
x14	-0.0675	0.041	-1.654	0.098	-0.148	0.013
x15	0.0950	0.028	3.371	0.001	0.040	0.150
x16	-0.0027	0.008	-0.356	0.722	-0.017	0.012
x17	-0.1161	0.025	-4.601	0.000	-0.166	-0.067
x18	-0.0017	0.035	-0.049	0.961	-0.070	0.067
x19	-0.1996	0.068	-2.949	0.003	-0.333	-0.067

Figure 4.21: Ordinary Least Square (OLS) Summary: Random Node Sampling and Spectral features Embedding for all features

OLS Regression Results

Dep. Variable:	distances	**R-squared:**	0.786
Model:	OLS	**Adj. R-squared:**	0.783
Method:	Least Squares	**F-statistic:**	309.3
Date:	Wed, 28 Apr 2021	**Prob (F-statistic):**	4.43e-274
Time:	23:32:56	**Log-Likelihood:**	419.19
No. Observations:	854	**AIC:**	-816.4
Df Residuals:	843	**BIC:**	-764.1
Df Model:	10		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	0.5465	0.005	106.970	0.000	0.536	0.557
x1	0.0222	0.015	1.440	0.150	-0.008	0.053
x2	0.0644	0.011	5.638	0.000	0.042	0.087
x3	0.0554	0.007	7.722	0.000	0.041	0.069
x4	-0.0273	0.012	-2.250	0.025	-0.051	-0.003
x5	-0.1584	0.013	-11.909	0.000	-0.184	-0.132
x6	-0.2954	0.019	-15.187	0.000	-0.334	-0.257
x7	-0.5448	0.058	-9.377	0.000	-0.659	-0.431
x8	1.1227	0.068	16.426	0.000	0.989	1.257
x9	0.0763	0.024	3.198	0.001	0.029	0.123
x10	-0.1183	0.023	-5.130	0.000	-0.164	-0.073

Omnibus:	14.443	**Durbin-Watson:**	2.038
Prob(Omnibus):	0.001	**Jarque-Bera (JB):**	14.129
Skew:	0.284	**Prob(JB):**	0.000855
Kurtosis:	2.725	**Cond. No.**	40.6

Figure 4.22: Ordinary Least Square(OLS) Summary: Random Node Sampling and Spectral features Embedding for Significant Features

OLS Regression Results

Dep. Variable:	distances	**R-squared:**	0.872
Model:	OLS	**Adj. R-squared:**	0.869
Method:	Least Squares	**F-statistic:**	293.7
Date:	Thu, 29 Apr 2021	**Prob (F-statistic):**	0.00
Time:	08:49:13	**Log-Likelihood:**	609.73
No. Observations:	839	**AIC:**	-1179.
Df Residuals:	819	**BIC:**	-1085.
Df Model:	19		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	0.4955	0.004	120.816	0.000	0.487	0.504
x1	0.0694	0.021	3.345	0.001	0.029	0.110
x2	-0.0018	0.020	-0.094	0.926	-0.040	0.036
x3	-0.0318	0.016	-2.040	0.042	-0.062	-0.001
x4	2.9029	0.260	11.170	0.000	2.393	3.413
x5	-0.0169	0.040	-0.425	0.671	-0.095	0.061
x6	-0.0178	0.023	-0.778	0.437	-0.063	0.027
x7	0.0465	0.008	5.542	0.000	0.030	0.063
x8	-0.0083	0.014	-0.591	0.555	-0.036	0.019
x9	-0.0091	0.006	-1.444	0.149	-0.022	0.003
x10	-0.8653	0.083	-10.395	0.000	-1.029	-0.702
x11	-0.4738	0.027	-17.560	0.000	-0.527	-0.421
x12	-0.3409	0.224	-1.520	0.129	-0.781	0.099
x13	-1.5176	0.419	-3.620	0.000	-2.340	-0.695
x14	-0.1409	0.017	-8.123	0.000	-0.175	-0.107
x15	-0.0037	0.017	-0.211	0.833	-0.038	0.031
x16	0.0498	0.010	4.934	0.000	0.030	0.070
x17	-0.0743	0.018	-4.117	0.000	-0.110	-0.039
x18	0.1316	0.044	2.970	0.003	0.045	0.219
x19	0.3835	0.311	1.235	0.217	-0.226	0.993

Figure 4.23: Ordinary Least Square(OLS) Summary: Random Edge Sampling and Graph2Vec Embedding for all features.

OLS Regression Results

Dep. Variable:	distances	**R-squared:**	0.862
Model:	OLS	**Adj. R-squared:**	0.860
Method:	Least Squares	**F-statistic:**	469.1
Date:	Thu, 29 Apr 2021	**Prob (F-statistic):**	0.00
Time:	09:29:19	**Log-Likelihood:**	582.28
No. Observations:	839	**AIC:**	-1141.
Df Residuals:	827	**BIC:**	-1084.
Df Model:	11		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	0.4920	0.004	116.947	0.000	0.484	0.500
x1	0.0570	0.015	3.681	0.000	0.027	0.087
x2	-0.0357	0.015	-2.314	0.021	-0.066	-0.005
x3	3.2022	0.202	15.877	0.000	2.806	3.598
x4	0.0398	0.008	4.877	0.000	0.024	0.056
x5	-0.9825	0.060	-16.259	0.000	-1.101	-0.864
x6	-0.4837	0.023	-20.993	0.000	-0.529	-0.438
x7	-1.6453	0.128	-12.852	0.000	-1.897	-1.394
x8	-0.1624	0.013	-12.976	0.000	-0.187	-0.138
x9	0.0464	0.007	6.260	0.000	0.032	0.061
x10	-0.0781	0.009	-8.987	0.000	-0.095	-0.061
x11	0.1140	0.015	7.690	0.000	0.085	0.143

Omnibus:	68.321	**Durbin-Watson:**	1.970
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	118.232
Skew:	0.560	**Prob(JB):**	2.12e-26
Kurtosis:	4.459	**Cond. No.**	140.

Figure 4.24: Ordinary Least Square (OLS) Summary: Random Edge Sampling and Graph2Vec Embedding for Significant Features.

OLS Regression Results

Dep. Variable:	distances	**R-squared:**	0.555
Model:	OLS	**Adj. R-squared:**	0.545
Method:	Least Squares	**F-statistic:**	53.79
Date:	Thu, 29 Apr 2021	**Prob (F-statistic):**	7.98e-130
Time:	10:29:22	**Log-Likelihood:**	631.06
No. Observations:	840	**AIC:**	-1222.
Df Residuals:	820	**BIC:**	-1127.
Df Model:	19		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	0.1749	0.004	43.706	0.000	0.167	0.183
x1	-0.0599	0.018	-3.376	0.001	-0.095	-0.025
x2	0.0100	0.018	0.541	0.588	-0.026	0.046
x3	0.1139	0.015	7.400	0.000	0.084	0.144
x4	-1.7188	0.247	-6.962	0.000	-2.203	-1.234
x5	-0.0239	0.037	-0.646	0.519	-0.097	0.049
x6	-0.0622	0.021	-2.965	0.003	-0.103	-0.021
x7	0.0498	0.008	5.864	0.000	0.033	0.067
x8	0.0214	0.014	1.537	0.125	-0.006	0.049
x9	-0.0086	0.005	-1.769	0.077	-0.018	0.001
x10	0.5228	0.078	6.674	0.000	0.369	0.677
x11	-0.0005	0.026	-0.017	0.986	-0.052	0.051
x12	-1.1298	0.204	-5.525	0.000	-1.531	-0.728
x13	1.3248	0.412	3.218	0.001	0.517	2.133
x14	0.1646	0.016	10.062	0.000	0.133	0.197
x15	-0.0566	0.017	-3.410	0.001	-0.089	-0.024
x16	0.0264	0.009	2.916	0.004	0.009	0.044
x17	0.0324	0.017	1.897	0.058	-0.001	0.066
x18	0.0004	0.040	0.010	0.992	-0.079	0.080
x19	1.0559	0.297	3.550	0.000	0.472	1.640

Figure 4.25: Ordinary Least Square(OLS) Summary: Random Edge Sampling and SF Embedding for all features

OLS Regression Results

Dep. Variable:	distances	**R-squared:**	0.582
Model:	OLS	**Adj. R-squared:**	0.576
Method:	Least Squares	**F-statistic:**	96.09
Date:	Thu, 29 Apr 2021	**Prob (F-statistic):**	1.16e-147
Time:	10:45:16	**Log-Likelihood:**	663.15
No. Observations:	840	**AIC:**	-1300.
Df Residuals:	827	**BIC:**	-1239.
Df Model:	12		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	0.1740	0.004	45.414	0.000	0.166	0.181
x1	-0.0580	0.016	-3.561	0.000	-0.090	-0.026
x2	0.0826	0.007	11.601	0.000	0.069	0.097
x3	-1.6708	0.150	-11.153	0.000	-1.965	-1.377
x4	-0.0265	0.010	-2.632	0.009	-0.046	-0.007
x5	0.0473	0.007	7.197	0.000	0.034	0.060
x6	0.5236	0.063	8.327	0.000	0.400	0.647
x7	-1.1872	0.189	-6.276	0.000	-1.559	-0.816
x8	1.1516	0.279	4.134	0.000	0.605	1.698
x9	0.1660	0.011	14.767	0.000	0.144	0.188
x10	-0.0329	0.008	-4.180	0.000	-0.048	-0.017
x11	0.0164	0.007	2.207	0.028	0.002	0.031
x12	1.2347	0.227	5.430	0.000	0.788	1.681

Omnibus:	83.678	**Durbin-Watson:**	2.052
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	175.099
Skew:	0.598	**Prob(JB):**	9.50e-39
Kurtosis:	4.890	**Cond. No.**	217.

Figure 4.26: Ordinary Least Square(OLS) Summary: Random Edge Sampling and SF Embedding for Significant Features

Table 4.6 gives the summary of all the the figures from 4.19 to 4.26 summary tables. A point

to note is that the Adjusted R-Squared we get is very similar to *10-fold cross-validation* using cross_val_score function from scikit-learn library [16].

Sampling Method	**Embedding Method**	**F-Statistic**	**Adjusted R-Squared**
Random Node Sampling	Graph2Vec	353.6	0.811
	Spectral Features	309.3	0.78
Random Edge Sampling	Graph2Vec	469.1	0.86
	Spectral Features	96.09	0.58

Table 4.6: Summary of F-Statistic and R-Squared values for Multi-domain Network Dataset. It shows OLS Summary for all embedding and sampling methods for all significant features of regression model.

Performance on Validation Datasets:

The OLS summary tables from figures 4.19 to 4.26 show results of how model is performed on the test data. To get a general perspective of how model performs on the completely new dataset, we decided to try on a new network that the model has not seen before. We call this dataset as validation dataset. To make it more realistic, we validate the model on a larger dataset than on which the model was trained on.

We used the Astrophysics dataset as our validation dataset(page 15). Figures 4.27 to 4.30 show the graph of actual similarity versus predicted similarity on the validation datasets for Random Node Sampling $\&$ Graph2Vec Embedding, Random Node Sampling $\&$ Spectral Features Embedding, Random Edge Sampling $\&$ Graph2Vec Embedding, and Random Edge Sampling $\&$ Spectral Features Embedding.

Figure 4.27 shows that our model predicts the values accurately. At 40% and 50% sampling, the values are very close. We think the reason for it could be as we got maximum similarity for

40% sampling from the experiment in section 4.1.

Figure 4.28 does not give very good results, but for 30% and 40% sampling the error between prediction and actual is less.

Figure 4.29 gives good results for lower sampling sizes but not for higher sampling sizes. The error increases after 30-40% sampling.

Figure 4.30 does not give good results because the training data does not have high similarities. As per figure 4.5, the highest similarity is around 0.2 which is less, so our model predicts similarities around that range but actual similarities are below 0.5. Thus performance of model on validation dataset depends on variation of values on dataset being validated. We can also see from performance of validation and test datasets that Spectral Features Embedding does not work well for Random Edge Sampling.

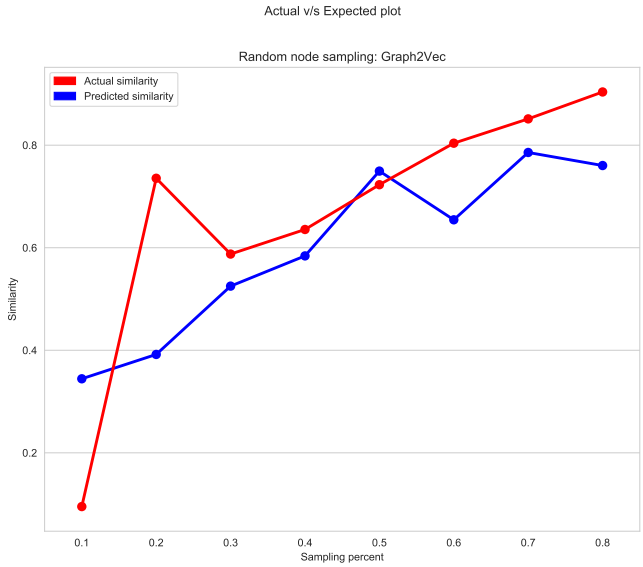


Figure 4.27: Actual versus Predicted for Astrophysics Dataset on Cross-Domain Regression Model: Random Node Sampling and Graph2vec embedding

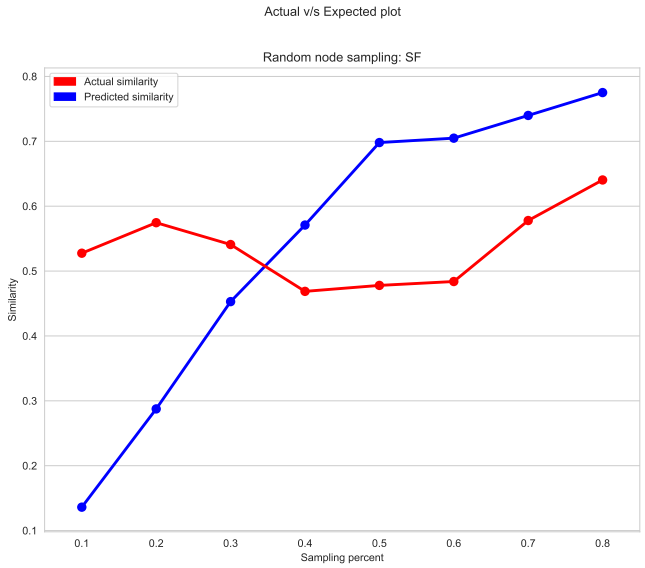


Figure 4.28: Actual versus Predicted for Astrophysics Dataset on Cross-Domain Regression Model: Random Node Sampling and Spectral Features embedding

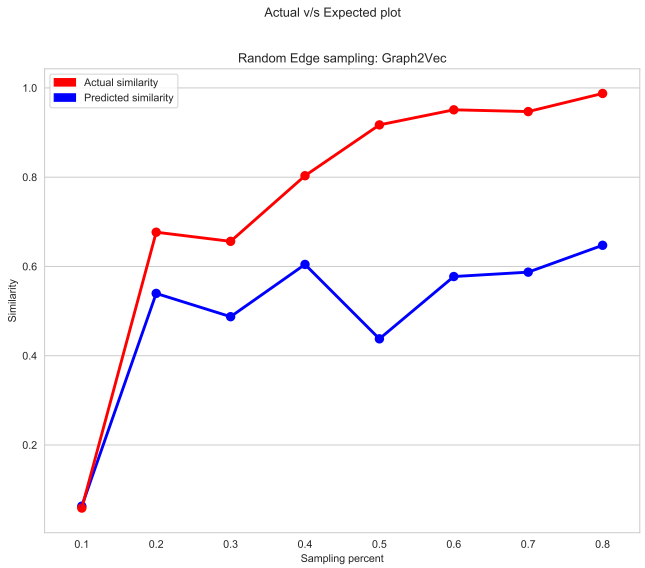


Figure 4.29: Actual versus Predicted for Astrophysics Dataset on Cross-Domain Regression Model: Random Edge Sampling and Graph2vec embedding

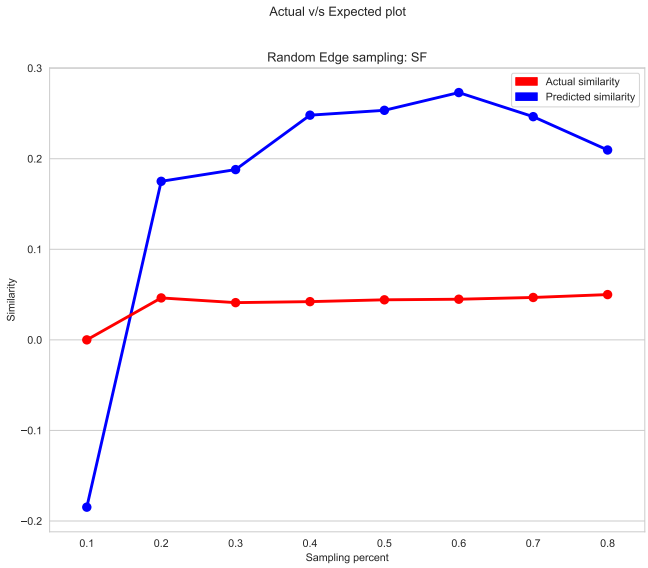


Figure 4.30: Actual versus Predicted for Astrophysics Dataset on Cross-Domain Regression Model: Random Edge Sampling and Spectral Features embedding

4.3 Stability of Embedding Methods

This section introduces stability from two papers and then provides our definition and stability analysis on the embedding methods.

4.3.1 Introduction

As we analyzed the two embedding methods for two sampling methods, we observe same embedding method behaves differently for different sampling methods. Hence, we want to go further to see if we can decide on the stability of embedding methods. Our prior analysis shows embedding methods also depend on the sampling method, so we define the stability of the embedding method for a particular sampling method. We build our idea of stability based on clustering stability [27]. The paper focuses on model selection based on clustering stability of K-means algorithm. It attempts to find K which generates stable clusters. We are more interested in the approach that the Von Luxburg takes to decide stability. Another approach is by the Schumacher et al. [24], where he calculates the stability of node embeddings using

two approaches. First, they consider geometric stability where they consider measures such as aligned cosine similarity, k-NN Jaccard Similarity, and Second-Order Cosine Similarity. Second, they studied Downstream Stability which is defined as (in)stability induced by embeddings on downstream classifiers. They perform experiments with GraphSAGE, HOPE, LINE, node2vec, and SDNE embeddings on synthetic and real-world graphs. They conclude HOPE results in constant embeddings while GraphSage appears to be volatile. For downstream stability experiments, overall classification was robust but individual node classification caused variations due to embeddings. They also study effect of graph properties such as size and density on stability of node embeddings for synthetic graphs and conclude that graph properties have small influence on stability but is overshadowed by choice of embedding algorithm. This paper is not directly related to our work but is included due to its relation to stability of embeddings.

4.3.2 Analysis

We define the stability of the embedding method as follows: *"Given a subgraph S taken from Graph G obtained by Sampling approach P and embedding method E, can we predict the similarity between S and G using just the graph properties of S, with a good amount of confidence? If yes, then we say the embedding method is stable."* In simple words, we define stability of the embedding method as the predictibility of the model as ultimately stability would also mean how much predictable the result is. So, we need to average the effect of all graph properties across all sampling sizes. But all the properties will not have the same scale, so need to do a weighted average. The Von Luxburg [27] also calculates the mean distance between clusterings and calculates stability for the clusters based on the mean distance. The lesser the mean distance, the more stable are clusters. In our case, we need to consider all properties including similarity, so we decide to consider R-Squared value of our regression model. R-squared is the coefficient of multiple determination for the multiple regression model.

R-Squared fits well here for two reasons: First, we use the same Linear Regression Model across all sampling and embedding methods. Second, it calculates the average with uniform

weights as defaulted by Scikit Learn. We are considering Scikit learn [16] R-Squared function as it is well defined with parameters. A point to note here is that the values given by OLS and by Scikit learn are the same with minor decimal differences which does not change the results.

We get R-Squared Values for the four models for multi-domain dataset as shown in Table 4.7:

Sampling Method	**Embedding Method**	
	Graph2Vec	**Spectral Features (SF)**
Random Node Sampling	0.81	0.78
Random Edge Sampling	0.87	0.57

Table 4.7: R-Squared Values for Graph2Vec and Spectral Features embeddings, for Random Node and Edge Sampling methods. These are R-Squared values of multi domain regression model

If we observe the values in the table 4.7 almost match the adjusted R-Square values from Figures 4.20, 4.22, 4.24 and 4.26.

From the table 4.7 we can say, Graph2Vec is a more stable algorithm than spectral features embedding. This shows Graph2Vec is stable embedding method overall. Another way to look at stability is that it directly relates to scalability. The stable method would be easily scalable. In the graph2vec [15] paper, the Narayanan et al. mention graph2vec is scalable due to the limited runs of skipgram training. Thus, it aligns with our definition of stability.

Chapter 5

CONCLUSION AND FUTURE WORK

In this chapter, we summarize the key findings of our experiments and methods, and provide directions for future work.

5.1 Conclusions

This section concludes the three points mentioned in contributions section. Please refer to page 3 to review contributions.

5.1.1 Minimum Sample Size

We introduced a new method for deciding the minimum sampling size for graph sampling based on distance between the graph embeddings of sample and that of the original graph. This gives an intuitive idea of minimum sample size to get ε-similarity to the original graph where the parameter ε differs for embedding and sampling methods. However, the graph is defined by various properties so we need to consider them along with the distances between the embeddings. Thus, same minimum sample will not work for all use cases or kinds of graphs. We solve this problem using Regression.

The table 5.1 gives summary for the minimum sample size as per sampling and embedding methods. Please refer to figures 4.2 to 4.5 for better visualization of overall results. As we can see, best results are obtained with Random Node Sampling which was also aligns with the finding by Leskovec et al. [11]. Higher similarity is obtained with Graph2Vec embedding.

Sampling Method	Embedding Method	Min. Sample Size	Similarity
Random Node Sampling	Graph2Vec	40%	80%
	Spectral Features	30%	60%
Random Edge Sampling	Graph2Vec	40%	65%
	Spectral Features	50%	25%

Table 5.1: Summary of Minimum Sample with respect to Sampling and Embedding method.

5.1.2 Regression

We characterize a graph using graph properties. When we know sampling size, we define graph properties at that sampling size. As we have the sample and the original graph, we can get embeddings which is used to calculate similarity. Hence, we develop a regression model which predicts similarity for a given sample based on the graph properties. We consider graph properties which captures structure and connectivity of graph from different perspectives. Experiments are conducted on domain specific and cross-domain graphs. Our observation states that domain specific gives nearly perfect results for that domain but fails on graphs from other domains. So, we add cross domain datasets and the model performs better for datasets of other domains.

5.1.3 Stability

The analysis in previous two steps will help in deciding minimum sample but we need some method that measures the goodness of the embedding method. So, we perform stability analysis based on R-Squared value which suggest Graph2Vec is overall stable for Random Node and Random Edge Sampling. The same is also illustrated in the summary table 5.2 of the results.

Sampling Method	Social Networks		Multi-domain Networks	
	Graph2Vec	**Spectral Features (SF)**	**Graph2Vec**	**Spectral Features (SF)**
Random Node Sampling	0.72	0.76	0.81	0.78
Random Edge Sampling	0.70	0.18	0.86	0.58

Table 5.2: Summary of the results of our experiments. The numbers denote Adjusted R-Squared values for Linear Regression on both datasets.

5.2 Future Work

Our experiments and analysis led to interesting usecases and applications. We discuss them in the subsections below:

5.2.1 Directed and Weighted Graphs

In this thesis, we focus on unweighted and undirected graphs. This approach can be extended to weighted and directed graphs. Along with additional properties for directed graphs, we also need to think about handling the weighted edges. Many complex networks, especially, social networks can also be represented using **hypergraphs**. Hypergraphs are more powerful and reveal information that is beyond simple node-edge relationship. It would be interesting to see if the information revealed by hypergraphs can be useful in finding out good samples and good embeddings.

5.2.2 Dynamic Graphs

We consider static graphs so we have embedding of original graph which always stays constant. For dynamic graphs, there is a possibility, same minimum sampling size will not work as graph evolves. To overcome this possibility, we can find minimum sampling size for a graph at time T. In addition, sampling and embedding methods also needs to be selected so that they are not

sensitive to evolution of graphs.

5.2.3 Other Similarity Measures

We consider Euclidean distance as the similarity measure. However, Euclidean distance comes with its limitations not giving correct results for high dimensional spaces, i.e. suffering from *curse of dimensionality*. For SF embedding, we have the highest similarity as 0.20, which might not be the real similarity. If we use a different metric, the similarity might give even higher value. Thus other similarity measures need to be explored. Using different similarity measure will also change the minimum sampling size. So, it will be interesting to further go ahead and see if there is a relation between similarity measures and minimum sampling sizes.

5.2.4 Mapping Between Minimum Sample Across Embeddings

In many studies of graphs, the approach or method in the study samples the graph using different sampling methods. To compare different methods, we need to know for example, $x\%$ of node sampling on one graph maps to $y\%$ of edge sampling on the same graph. The knowledge of this mapping between different samples for the same embedding method will help in comparing various studies of graphs in future.

5.2.5 Varying embedding Size

We currently consider the default embedding size as 128 provided by karateclub library [22]. The experiments can be done by varying embedding sizes and see if the embedding size impacts the minimum sampling size.

5.2.6 Effect of Sampling and Embedding on Graph Properties

In this thesis, graph properties for a particular sample is given to regression model as input. An interesting question to answer would be how different graph properties change with respect to sample? We define stability in terms of R-Squared value which considers the graph properties.

Some research directions in these areas include answering questions such as what is the behavior of properties in stable versus less stable embedding methods? or, Do the graph properties contribute significantly to the stability of embeddings?

BIBLIOGRAPHY

[1] Y. Bai, H. Ding, S. Bian, T. Chen, Y. Sun, and W. Wang. Simgnn: A neural network approach to fast graph similarity computation. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*, pages 384–392, 2019.

[2] L. Buitinck, G. Louppe, M. Blondel, F. Pedregosa, A. Mueller, O. Grisel, V. Niculae, P. Prettenhofer, A. Gramfort, J. Grobler, R. Layton, J. VanderPlas, A. Joly, B. Holt, and G. Varoquaux. API design for machine learning software: experiences from the scikit-learn project. In *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, pages 108–122, 2013.

[3] F. Chen, Y.-C. Wang, B. Wang, and C.-C. J. Kuo. Graph representation learning: a survey. *APSIPA Transactions on Signal and Information Processing*, 9, 2020.

[4] N. de Lara and E. Pineau. A simple baseline algorithm for graph classification. *arXiv preprint arXiv:1810.09155*, 2018.

[5] L. A. Goodman. Snowball sampling. *The annals of mathematical statistics*, pages 148–170, 1961.

[6] A. Hagberg, P. Swart, and D. S Chult. Exploring network structure, dynamics, and function using networkx. Technical report, Los Alamos National Lab.(LANL), Los Alamos, NM (United States), 2008.

[7] D. D. Heckathorn. Respondent-driven sampling: a new approach to the study of hidden populations. *Social problems*, 44(2):174–199, 1997.

[8] P. Hu and W. C. Lau. A survey and taxonomy of graph sampling, 2013.

[9] S. Jin and R. Zafarani. The spectral zoo of networks: Embedding and visualizing networks with spectral moments. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1426–1434, 2020.

[10] M. Kurant, M. Gjoka, Y. Wang, Z. W. Almquist, C. T. Butts, and A. Markopoulou. Coarse-grained topology estimation via graph sampling. In *Proceedings of the 2012 ACM workshop on Workshop on online social networks*, pages 25–30, 2012.

[11] J. Leskovec and C. Faloutsos. Sampling from large graphs. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 631–636, 2006.

[12] J. Leskovec and A. Krevl. SNAP Datasets: Stanford large network dataset collection. `http://snap.stanford.edu/data`, June 2014.

[13] J. Leskovec, J. Kleinberg, and C. Faloutsos. Graph evolution: Densification and shrinking diameters. *ACM transactions on Knowledge Discovery from Data (TKDD)*, 1(1):2–es, 2007.

[14] F. Morstatter, J. Pfeffer, H. Liu, and K. Carley. Is the sample good enough? comparing data from twitter's streaming api with twitter's firehose. In *Proceedings of the International AAAI Conference on Web and Social Media*, volume 7, 2013.

[15] A. Narayanan, M. Chandramohan, R. Venkatesan, L. Chen, Y. Liu, and S. Jaiswal. graph2vec: Learning distributed representations of graphs, 2017.

[16] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[17] R. Pepper. On the annihilation number of a graph. In *Recent Advances In Electrical Engineering: Proceedings of the 15th American Conference on Applied Mathematics*, pages 217–220, 2009.

[18] M. Ripeanu, I. Foster, and A. Iamnitchi. Mapping the gnutella network: Properties of large-scale peer-to-peer systems and implications for system design. *arXiv preprint cs/0209028*, 2002.

[19] B. Rozemberczki and R. Sarkar. Characteristic Functions on Graphs: Birds of a Feather, from Statistical Descriptors to Parametric Models. In *Proceedings of the 29th ACM International Conference on Information and Knowledge Management (CIKM '20)*, page 1325–1334. ACM, 2020.

[20] B. Rozemberczki, C. Allen, and R. Sarkar. Multi-scale attributed node embedding, 2019.

[21] B. Rozemberczki, R. Davies, R. Sarkar, and C. Sutton. Gemsec: Graph embedding with self clustering. In *Proceedings of the 2019 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2019*, pages 65–72. ACM, 2019.

[22] B. Rozemberczki, O. Kiss, and R. Sarkar. Karate Club: An API Oriented Open-source Python Framework for Unsupervised Learning on Graphs. In *Proceedings of the 29th ACM International Conference on Information and Knowledge Management (CIKM '20)*, page 3125–3132. ACM, 2020.

[23] B. Rozemberczki, O. Kiss, and R. Sarkar. Little Ball of Fur: A Python Library for Graph Sampling. In *Proceedings of the 29th ACM International Conference on Information and Knowledge Management (CIKM '20)*, page 3133–3140. ACM, 2020.

[24] T. Schumacher, H. Wolf, M. Ritzert, F. Lemmerich, J. Bachmann, F. Frantzen, M. Klabunde, M. Grohe, and M. Strohmaier. The effects of randomness on the stability of node embeddings. *arXiv preprint arXiv:2005.10039*, 2020.

[25] S. Seabold and J. Perktold. statsmodels: Econometric and statistical modeling with python. In *9th Python in Science Conference*, 2010.

[26] J. Ugander, B. Karrer, L. Backstrom, and C. Marlow. The anatomy of the facebook social graph. *arXiv preprint arXiv:1111.4503*, 2011.

[27] U. Von Luxburg. Clustering stability: an overview. 2010.

[28] C. Wang, C. Wang, Z. Wang, X. Ye, and P. S. Yu. Edge2vec: Edge-based social network embedding. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 14(4):1–24, 2020.

[29] R. Zafarani, M. A. Abbasi, and H. Liu. *Social Media Mining: An Introduction*. Cambridge University Press, New York, NY, USA, 2014. ISBN 1107018854, 9781107018853.

APPENDIX A

RAW DATA

The code for all the experiments in this thesis can be found on Github:
https://github.com/apurvamulay/Embedding-Similarity-and-Stability.

APPENDIX B

VITA

Apurva Mulay

asmulay@syr.edu | linkedin.com/in/apurva-mulay/ | github.com/apurvamulay

Education

Syracuse University, Syracuse, NY Aug. 2019 – May 2021
Master of Science,Computer Science, Thesis (Machine Learning) *GPA: 3.72/4*

Courses: Analysis of Algorithms, Intro to Data Science, Analytical Data Mining, Social Media Mining, NLP

Pune University, Pune, India Jul. 2011 – May 2015
Bachelor of Engineering, Computer Engineering *GPA: 3.6/4*

Experience

Research Assistant Apr. 2020 – Present
Syracuse University *Syracuse, United States*

- Researching on Similarity and Stability of Graph embedding methods in thesis advised by Dr. Reza Zafarani - Networkx, Stanford SNAP datasets, Linear Regression, Decision Tree Regressors
- Analyzing COVID-19 related news on Twitter to derive patterns during pandemic and predict misinformation
- Collaborated on developing *first* fake news multimodal dataset using REST API, news API and performed data cleaning, preprocessing, data normalization, and feature engineering with Python3, Pandas, matplotlib, Sklearn

Software Engineering Intern Jun 2020 – Aug 2020
Xandr, AT&T *New York City, United States*

- Visualized 10GB data in line graphs, pie charts and bar graphs to help publishers in decision making by reducing analysis time by 70% in D3.js, Javascript, and Vertica Database
- Utilized GraphQL APIs to improve page load time performance by 50% and automated deployment using Kubernetes
- Created core graph library to be used across projects and presented it in Lightning talks
- Conducted informational interviews with Data Scientists, and Product Managers to gain domain knowledge

Software Engineer July 2015 – Aug 2019
Mediaocean *Pune, India*

- Implemented new features to serve 58 major advertising agencies, across US, UK and Australia using Java, Javascript, ReactJS, HTML5 and CSS3. Unit tested using Jest and Enzyme.
- Analyzed and reported advertising campaign performance for over 50 campaigns using MYSQL
- Dockerized projects to reduce external library version dependency and integrated them using Jenkins
- Worked on proof of concept projects on recommendation systems, chatbot and classification to apply Machine learning to advertising domain

Publications

ReCOVery: A multimodal dataset for COVID-19 news credibility research

- Crawled news articles from **22** reliable and **38** unreliable news websites with corresponding 140k tweets
- Co-authored dataset paper ReCOVery to be analyzed by researchers and accepted into **CIKM 2020 Conference**
- **Technologies:** Scikit-Learn, SVM, Pandas, Seaborn, Matplotlib, NLP, Twitter Advanced and Streaming APIs, Text-CNN, RNN, Git, Beautifulsoup, NLTK, Naive Bayes

Projects

Retweet Prediction | *Python, Regression, Sklearn* Nov 2020 – Dec 2020

- Performed prediction of retweets count for TweetsCOV19 dataset using Linear, Ridge and LASSO regression
- Inspired by CIKM 2020 Analyticup competition and achieved MSLE of 0.2 using Ensemble methods

Spread of COVID-19: Analysis and Prediction | *LIWC, TFIDF* Apr 2020 – May 2020

- Conducted time series forecasting to predict new cases with MSE of 0.17 using ARIMA
- Performed sentiment analysis on effect of government policies on 10k tweets

Music Taste Analysis using Spotify API | *Python, Classification* Nov 2019 – Dec 2019

- Predicted music taste analysis of Spotify user across multilingual songs and genre using Python, Sklearn, K-Fold cross validation, regularized logistic regression, decision trees and ensemble methods
- Collected data using Spotify REST API for 5 user profiles
- Achieved F1 Score of 0.90 and accuracy of 91% with Random Forest classifier and visualized using Tableau

Mediaocean Hackathon: Chatbot | *Microsoft Azure* May 2018 – May 2018

- Developed a chatbot application with active learning to reduce turnaround time for support tickets by 70% using Microsoft Azure, Apache OpenNLP, and React.js for visualization
- *Award:* 1st Place Winner out of 30 teams

Achievements and Awards

- Secured First runner-up position at **Google Tech Challenge** at Syracuse University where over 15 teams participated
- Recognized for distinguished performance at work and presented with the **Brand Value Award - Initiative (Global award)** for initiating React.js architecture and Machine Learning presentations at Mediaocean