北京郵電大學

Queen Mary University of London

Undergraduate Project Report
2019/20

The implementation of temporal graph management system

Name:	Weiqiang Yu
School:	International School
Class:	2016215113
QM Student No.:	161196754
BUPT Student No.:	2016213346
Programme:	e-Commerce Engineering with Law

Date: 30-04-2020

Table of Contents

Abstract

A temporal graph, also known as a dynamic graph, is a graph that changes through time. The exploration of such temporal graphs is a fundamental problem for graph analysis. Currently, there are a few systems that simply display the basic statistics of dynamic temporal graphs. A prominent example is SNAP (Stanford Network Analysis Project), a highly famous website in this field. However, it does not provide user-friendly services for users to further explore and understand the properties of the graph. To resolve this issue, we present a temporal graph management system, which displays various information about the dynamic temporal network, including both basic statistics and deep insights. With the help of this system, users can easily explore the properties of graphs, and mine structural patterns (e.g., community) on top of the temporal graphs. By adopting the framework of Apollo, this system could predict graph operator outputs efficiently and accurately. At the aspect of structural mining, the system supports community detection methods, including Louvain and OLCPM, to extract the community structure. Furthermore, this project also implements some community search algorithm to analyse the communities that a certain vertex belongs to in different periods. In addition, this project is designed as a website so that user can easily access the system through the Internet.

Keywords: temporal network, community structure, Apollo framework

摘要

时间图（temporal graph），也称为动态图，是随时间变化的图。探索时间图是图分析的一个基本问题。当前，一些系统只显示动态时间图的基本统计信息。一个杰出的例子是 SNAP（Stanford Network Analysis Project），它是该领域一个非常有名的网站。但是，它并没有提供友好的服务让用户进一步探索和理解图的属性。为了解决此问题，我们提出了一个时间图管理系统，该系统展示有关动态时间网络的各种信息，包括基本统计信息和深刻见解。借助该系统，用户可以轻松浏览图的属性，并在时间图上挖掘结构模式（例如，社区）。通过采用 Apollo 框架，我们的系统可以高效，准确地预测图运算符的输出。在结构挖掘方面，该系统支持 Louvain 和 OLCPM 社区检测方法来提取社区结构。此外，该项目还实现了一些社区搜索算法来分析顶点在不同时期所属的社区。我们的系统以网站的形式呈现，以便用户可以轻松地通过互联网访问该系统。

关键词：时间网络，社区结构，Apollo 框架

Chapter 1: Introduction

1.1 The point of the project

Temporal networks are networks where edges have timestamps. They are useful to analyse how a system evolves through time. The exploration of such temporal graphs is a fundamental problem for graph analysis. Currently, only a few systems display the information of temporal graphs. Stanford Network Analysis Project (SNAP) is a pioneer in this area. SNAP is a network analysis and graph mining library. It is mainly used for the research of complex networks. It contains a great many of large network datasets, including temporal networks. However, SNAP only shows some basic network information and does not provide user-friendly services for users to interact with the system. Considering this limitation of SNAP, this project was designed to build a temporal graph management system that allows users to explore the properties of temporal graphs freely.

1.2 What has been achieved

Currently, a web-based temporal graph management system has been designed. Figure 1 displays the system architecture of this project. This project used Django as the web framework and was deployed on Linux by Apache. People could get access to our system through Internet with the IP address 152.136.59.62 and the port 8080. This system contains a great deal of temporal network information, including metadata and statistical data. With powerful visualization tools, such as Plotly and D3.js, this system involves a variety of visualization components, including data distribution map, force-directed graph, heatmap and dataset space etc. Thus, users could easily analyse the dynamic graph and interact with the system freely. Additionally, through the adoption of Apollo framework, this system could predict graph operator output, like diameter, efficiently. Degree distribution methods are used to compare the similarity between graphs. Based on the calculated similarity matrix, k-Nearest-Neighbours (kNN) algorithm is employed to estimate the graph operator outputs. Furthermore, this system implements community detection and community search functions as well. Our system adopts OLCPM for local community detection, Louvain algorithm for global community detection, and local K-core-based method for community search. Now, users are capable of exploring the community structure of the network very efficiently. This project also supports upload and download components. Users could upload their own graph datasets, and then the system will process the data and generate corresponding data presentation pages. Users could also

download the original datasets in our website as well. Moreover, a specification document, which describes the system functions in detail, is also provided in the system. Finally, several tests for system functions were carried out, which could reveal the overall effect and time complexity of this system.

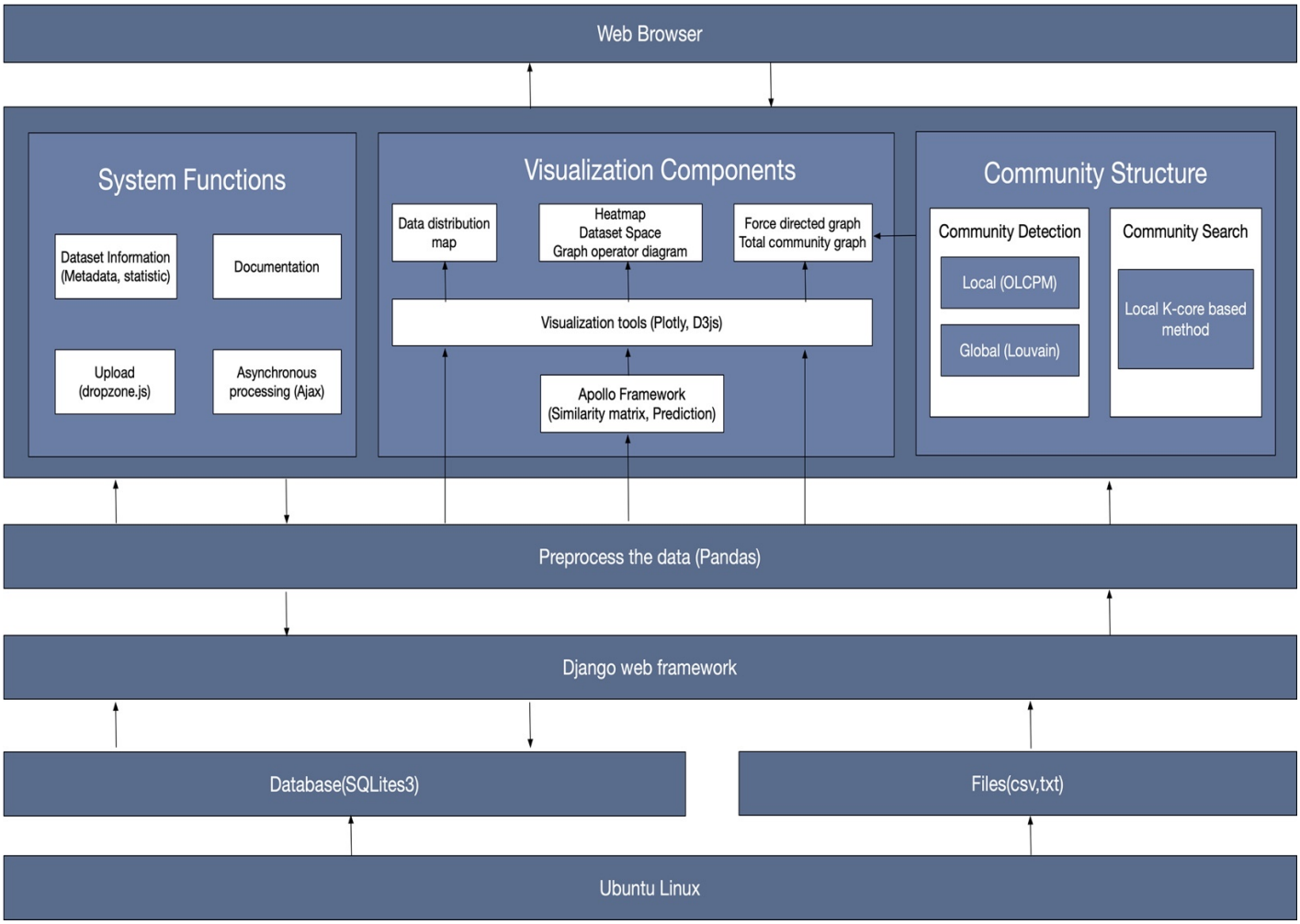

Figure 1 System Architecture

1.3 The structure of the report

This report includes five chapters.

Chapter 1: Introduction. This chapter shows the point of the project, states my contribution to this project and gives people an overall picture of the project.

Chapter 2: Background. This chapter introduces some useful relevant background knowledge. It briefly explains that concept of degree distribution and several commonly used techniques for comparing the degree distributions between graphs. Besides, this chapter also explains the idea of community structure, including local community definitions and global community definitions, so that people could follow the rest of report easily.

Chapter 3: Design and Implementation. This section first states the design of the web-based system, involving database design, system functions and performance optimization. Next, the implementation of visualization components is discussed, containing the functions and process for creating these visualization components. In addition, this chapter also elaborates the design and implementation of community detection and community search functions. It involves detailed algorithm ideas and steps for building these methods.

Chapter 4: Results and Discussion. In chapter four, the process and the outcome of the project are displayed. This section discusses multiple tests for this project, including tests for visualization components, community structure and the whole system. These well-designed tests are helpful to reveal the efficiency and effectiveness of this system.

Chapter 5: Conclusion and Further Work. This chapter reiterates the previous chapters, and draws a conclusion of the overall project. Besides, the further work of the project is also discussed here.

Chapter 2: Background

2.1 Degree Distribution

In graph theory and networks, degree refers to the number of connections between one point and other points in the network (graph). Degree distribution is the probability distribution of the number of degrees at each point in the network.

Multiple methods have been introduced to compare the degree distribution of graphs.

2.1.1 Kullback–Leibler divergence

KL divergence could measure the difference between two degree distributions: one is the true distribution of the data; the other is the theoretical distribution of the data.

$$D_{\mathrm{KL}}(P \parallel Q) = \sum_{x \in \mathcal{X}} P(x) \log\left(\frac{P(x)}{Q(x)}\right). \tag{1}$$

If two distributions p and q are the same, the KL divergence is 0.
However, KL divergence is asymmetrical and it does not satisfy the triangle inequality.

2.1.2 Jensen–Shannon divergence

JS divergence is another method to measure the similarity between degree distributions. It is a variant based on KL divergence, and resolves the asymmetry problem of KL divergence.

$$\begin{aligned} \mathrm{JSD}(P \parallel Q) &= \frac{1}{2} D(P \parallel M) + \frac{1}{2} D(Q \parallel M) \\ \text{where } M &= \frac{1}{2}(P + Q) \end{aligned} \tag{2}$$

JS divergence is symmetrical, and its value is between 0 and 1. If p is completely overlapped with q, the JS divergence is 0. If they do not intersect with each other, then the JS divergence is 1.

2.1.3 Bhattacharyya Coefficient

Bhattacharyya Coefficient is an approximate calculation of the overlap of two statistical samples. The BC coefficient can be used to measure the correlation between two groups of samples [1].

$$BC(p,q) = \sum_{x \in X} \sqrt{p(x)q(x)} \tag{3}$$

The range of BC coefficient is also between 0 and 1. Unlike JS divergence, if two distribution p and q are exactly the same, then BC coefficient is 1. If these two distributions do not overlap, then BC coefficient is 0.

In this project, JS divergence and BC coefficient are employed to compare the similarities between datasets. Both these two algorithms could generate the similarity matrix very quickly.

2.2 Community Structure

In reality, complex systems may involve neural network, social network etc. For these complex systems, complex networks could be used to represent them. A complex network is a network structure composed of a large number of nodes and edges with each other. Nodes represent the objects in the complex system, and edges represent the relationship between objects.

The communities in the network are usually composed of a collection of nodes. These network nodes usually have similar functions or similar properties. Therefore, communities are considered to help reveal the relationship between the network structure and functions.

Based on whether nodes connect to each other densely on a subgraph or on the whole network, we could define local and global community [2].

2.2.1 Local community definitions

As for local community, it is often related to the concept of subgraph cohesiveness and mutuality, like k-cliques and k-cores. Besides, it is possible to define local communities based on internal and external vertex and subgraph degrees. What's more, we could also define some local measure of community quality. Then, for a given subgraph, we could use our measure to quantify the degree of community-ness.

k-clique: A clique is a collection of vertices where any two vertices are connected, that is, a complete subgraph. A k-clique represents a complete subgraph with k nodes in the network. If a k-clique overlaps with another k-clique with k-1 nodes, then the two k-clique are connected. A set consisting of all connected k-clique is a k-clique community.

k-core: k-core defines a subgraph structure. In the subgraph, each vertex has at least k adjacent vertices. Hence, when k is a large number, k-core can be considered as a subgraph with close

connections between vertices, but the subgraph is not necessarily connected.

2.2.2 Global community definitions

In terms of global community, it is often involved with the concept of modularity.

Modularity is commonly used to measure the strength of network structures. The value of the modularity is primarily dependent on the network partition. It could be used to quantify the quality of network partition. When the value approaches to 1, the partition is better. Hence, it is feasible to obtain the optimal network partition by maximizing the modularity Q.

In this project, k-clique based local community detection was employed to extract overlapping communities. By maximizing the modularity, Louvain algorithm was implemented to identify global community. Furthermore, this project also utilized k-core-based local community search method to detect a maximal community that contains the given query node.

Chapter 3: Design and Implementation

3.1 A web-based temporal graph management system

3.1.1 System Function Design

Currently, a web-based temporal graph management system has been designed. Users could access to our system through the Internet anywhere and anytime. The web framework of this system is Django. Django is an open source, Python-based web application framework, which follows the design pattern of model-template-view (MTV). With Django, a high-quality, easy-to-maintain, database-driven applications could be created in minutes.

Figure 2 explains the framework of Django. Django supports SQLites3 as the database by default. The model file creates the database schema and the view file contains all the system functions. All the html files are stored in template folder, while other static files, such as css and js, are placed in the static folder. URL dispatcher could map the URL of the website to the function in view, which fetches the html in template.

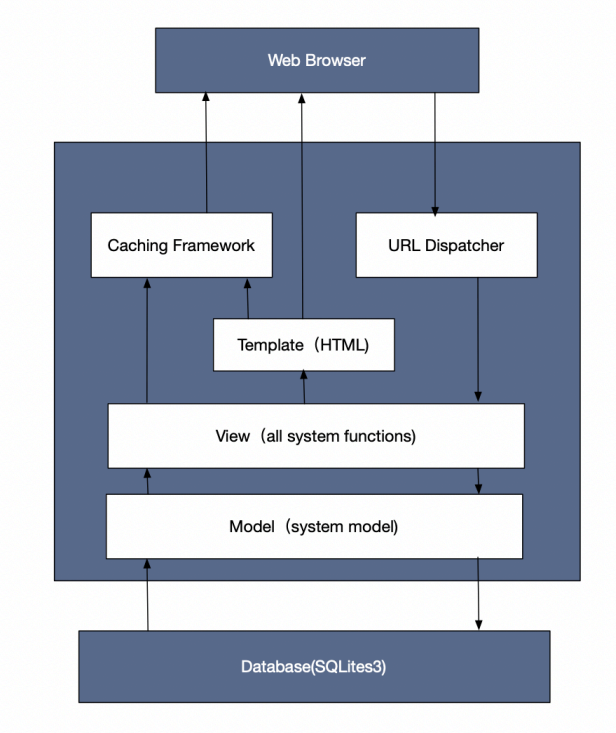

Figure 2 Django Framework

For this project, there are multiple pages displaying the dynamic temporal network information.

Home Page: For the home page, people could see the general information of the datasets, like network metadata and network statistical data. Network metadata involves network name and

network type. Network statistical data includes the number of nodes, temporal edges and static edges. Users could also download the data files here.

Data Presentation Page: On the data presentation page, more statistical data is presented, including time span, maximum degree and average degree. Users could see the network data preview in this page as well. Additionally, this page also includes several visualization components, involving data distribution map, heatmap and dataset space. Moreover, users could also explore the community structure by running community detection or community search methods here. Furthermore, predicting graph operator output is also feasible on this page.

Upload Page: Except existing datasets, users are also able to upload their own dataset files in our system. They first need to fill in the required form. Next, they need to select the dataset and submit the form. To design the upload component, dropzone.js was utilized. Dropzone.js is a very popular and powerful tool to achieve the upload function. Users could just drag and drop the file, then it will automatically transfer the file to the server. Besides, Dropzone.js also allows developers to place several limitations for uploading files. For example, in this project, users could only upload txt, csv files, and they could only upload one file at a time. In addition, the dataset file uploaded should at least contain three columns: SRC, DST and UNIXTS. Otherwise, this system would fail to process the submitted data.

Documentation Page: A detailed documentation page is created to instruct users to operate our website correctly. The documentation explains the main feature, file structure and application structure of the temporal graph management system so that users could have an overall picture of this system. Besides, the documentation also contains some description of system functions and method properties, such as default parameters. With such useful information, users could get a better understanding of this project.

404 Page and others: When users visit the pages that could not be found on the server, the system will automatically redirect to 404 page. Moreover, multiple error handling mechanisms were also implemented in the system to elegantly deal with the occurrence of error. For example, if a user enters an invalid time interval, the website will remind the user that their input is incorrect.

In order to enable more people to get access to the website, Apache2 and Virtualenv were adopted to deploy the project on Linux. And now, people could visit the system with the IP address 152.136.59.62 and port 8080.

Table 1 shows the data access interface of the website functions.

Table 1: data access interface of the website function

Function of the interface	Interface
Redirect to index page	def index(request)
Redirect to data presentation page	def document(request,type,id)
Redirect to upload page	def uppage(request)
Redirect to documentation page	def doc(request)
Redirect to 404 page	def handler404(request,exception):
Download one specific dataset	def download(request,filename):
Download all datasets in zip format	def zipdownload(request)
Upload one dataset	def upload_file(file)

3.1.2 Database Design

There are two kinds of data in our system. One is the raw data, the other is dataset information, such as metadata and statistical data.

The raw data is stored in files with txt or csv format. The system first utilized Pandas to read the files, and then applied some functions to obtain the statistical data.

As for dataset information, it is stored in the sqlite3 database, which is the default database of Django. Currently, three databases have been designed for displaying the temporal network information.

The first database is Datainfo, which stores the data of temporal networks. Table 2 illustrates the attributes and corresponding description of Datainfo database. It is noteworthy that there are two kinds of edges here: static edges and temporal edges. Temporal edges are edges with timestamps, while static edges dismiss the factor of time. For example, if there are two edges: one is from a to b with timestamp 0; the other one is from a to b with timestamp 1. Then there are two temporal edges. However, there is only one static edge since both these two edges are from a to b.

Table 2: Datainfo database table

Attribute	Description
Id	the id of the network
Name	the name of the network
Type	the type of the network, like directed, weighted, temporal network
Nodes	number of nodes in the network
Temporal Edges	number of temporal edges
Static Edges	number of static edges
Description	other details of the network

The second database is called Subinfo. It contains the information of each specific dataset. As shown in the table 3, there is a huge amount of data in this database, involving metadata (name, description), statistical data (max degree, min degree) and graph data (distribution, heatmap). In this way, it reduces the workload of computation, so users could have a better experience when they visit this website.

Table 3: Subinfo database table

Attribute	Description
Id	the id of the dataset
Name	the name of the dataset
TimeSpan	time period of the dataset in days
Nodes	number of nodes in the dataset
Temporal Edges	number of temporal edges
Static Edges	number of static edges
Description	other details of the network
Distribution	the distribution map of the dataset
Max degree	the maximum degree in the dataset
Min degree	the minimum degree in the dataset
Average degree	the average degree in the dataset
Max date	the final date of the dataset
Min date	the start date of the dataset
Heatmap	the heatmap of the dataset in different time
Dataspace	the dataset space of the dataset in different time
Total communities	the graph of total communities in different time
Force nodes	nodes in json format, also indicate which community nodes belong to
Force links	edges in json format
Community	the community extracted from the network by Louvain algorithm
Size	the size of the community
Prediction	the prediction graph of diameter

Last, Otherdata database was created to store the information of uploaded datasets. Thus, when users upload their own datasets, the system will first process the data they submitted, and then it stores useful information into this database. The structure of Otherdata database is similar to Subinfo.

3.1.3 Performance optimization

In order to give users a better experience of this system, two approaches were adopted to optimize the performance of the system. Figure 3 shows us the process of performance optimization.

The implementation of temporal graph management system

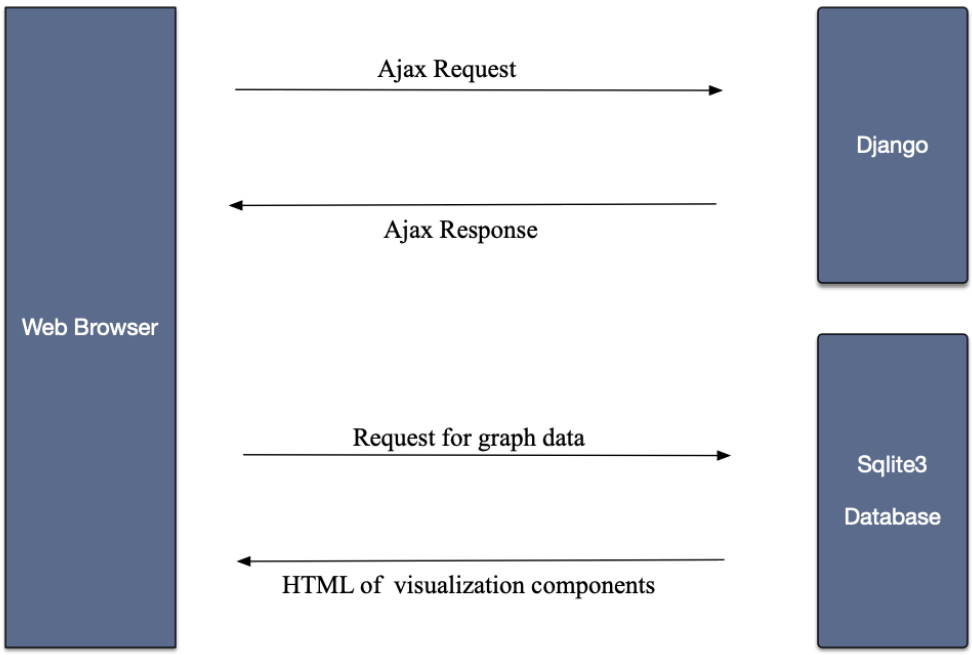

Figure 3 Performance Optimization Process

To begin with, when users make requests to the system, some functions may take several seconds. If the system chooses synchronous processing method, users have to wait until the response appears. Besides, if users submit a form, the whole page needs to be reloaded, which may cause extra waiting time. As a result, users may feel this system is quite user-unfriendly. To resolve this issue, ajax is employed to realize asynchronous processing. Now, when users make requests to the server, the webpage could be updated locally without reloading the existing page. What's more, while waiting for the response, users could navigate other part of the webpage to gain more information of the dataset. Moreover, dropzone.js also provides asynchronous service. Hence, when users upload their datasets, they could visit other pages without interfering the upload function.

Secondly, since data presentation page involves multiple visualization components, which may result in huge computational expense. Hence, when users visit data presentation page, it may take a while for this page to be generated. To speed up the page load, storing graph information into database is a great option. For example, to store the heatmap and dataset space into database, the system first calculated similarity matrix. Then it utilized Plotly to generate corresponding html and stored it into the database. Now, whenever users visit data presentation page, our website could directly retrieve the heatmap and dataset space from the database without a huge computation effort.

The implementation of temporal graph management system

Table 4 displays the data access interface of performance optimization.

Table 4: data access interface of performance optimization

Function of the interface	Interface
Receive ajax request and update data distribution map	def changeChart(request)
Receive ajax request and update force-directed graph	def changed3(request)
Receive ajax request and update heatmap and dataset space	def change_apollo(request)
Receive ajax request and update graph operator diagram	def prediction(request)
Calculate statistical information out of raw data	def preprocess(data,df)
Process the data and store it in the database	def upload(request)

3.2 A variety of visualization components

3.2.1 Data distribution map

Data distribution map allows users to observe the distribution of data in different time span. The system first divides the dataset by given granularity, like year. Next, it counts the amount of data (edges) in different time periods. Then it utilizes Plotly, an open-source, interactive graphing library for Python, to create the distribution map based on the calculated result.

3.2.2 Force-directed Graph

Force-directed graph can represent the many-to-many relationship between nodes. With the force-directed graph, users can observe the network architecture intuitively.

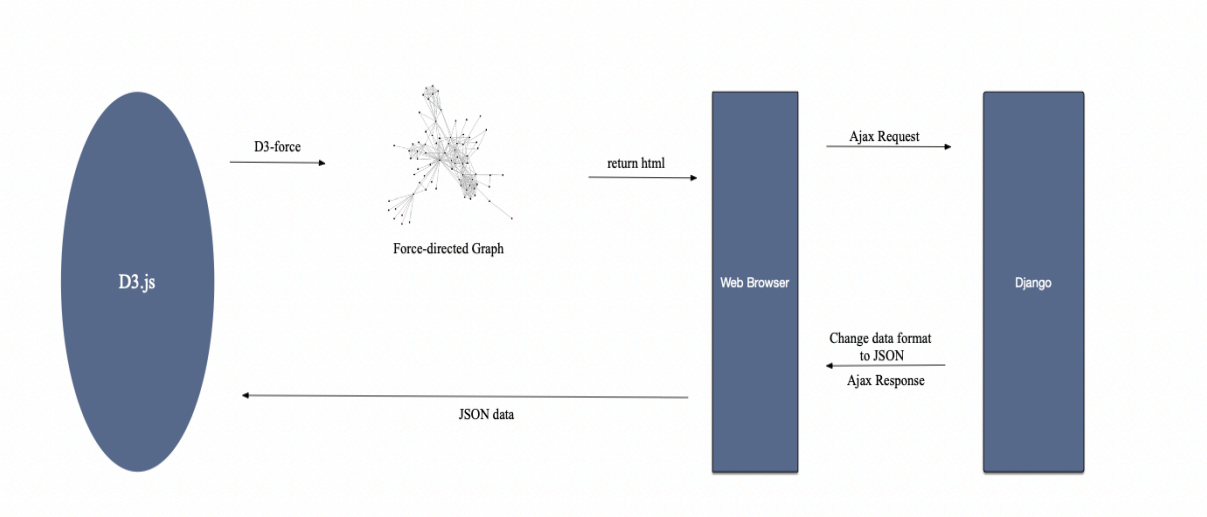

Figure 4 Force-directed Graph Procedure

Figure 4 states the procedure of force-directed graph. To draw the graph, the system first uses Pandas to extract the unique nodes in the dataset, and converts these nodes and all edges into

JSON format. Next, the system utilizes D3.js, a JavaScript library for visualizing data using web standards, to draw the graph. In order to represent all the nodes in the graph with different colours based on their communities, the system assigns "group" attribute to all the nodes. "Group" attribute shows which community nodes belong to. If two nodes have the same group value, then they share the same community, and they will have the same colour in the graph. However, since nodes could belong to multiple communities, it is infeasible to use "group" attribute for displaying overlapping communities. To address this problem, the system assigns an extra attribute, community, to these nodes. The "community" attribute stores all the communities this node belongs to. The system also assigns -1 to these nodes' group values. As a result, these nodes will have the colour in the graph and people will recognize them easily. To show the "community" attribute in the graph, the system appends "title" markup, and then D3.js will process these "title" markup. Now, when the mouse hovers over these nodes for a second, all communities these nodes belong to will appear.

3.2.3 Heatmap, Dataset Space and Graph Operator Prediction

Besides, inspired by the framework of Apollo [4], this system can create heatmap, dataset space and predict graph operator output. Apollo framework is based on the intuition that, for a given graph operator, similar graphs produce similar outputs. The main idea of Apollo could be summarized into three steps. First, based on some graph attributes, it generates the similarity matrix between graphs. Next, it applies visualization tools to generate heatmap and dataset space. Last, based on the calculated similarity matrix, it predicts graph operator outputs.

Figure 5 describes the whole procedure of Apollo Framework.

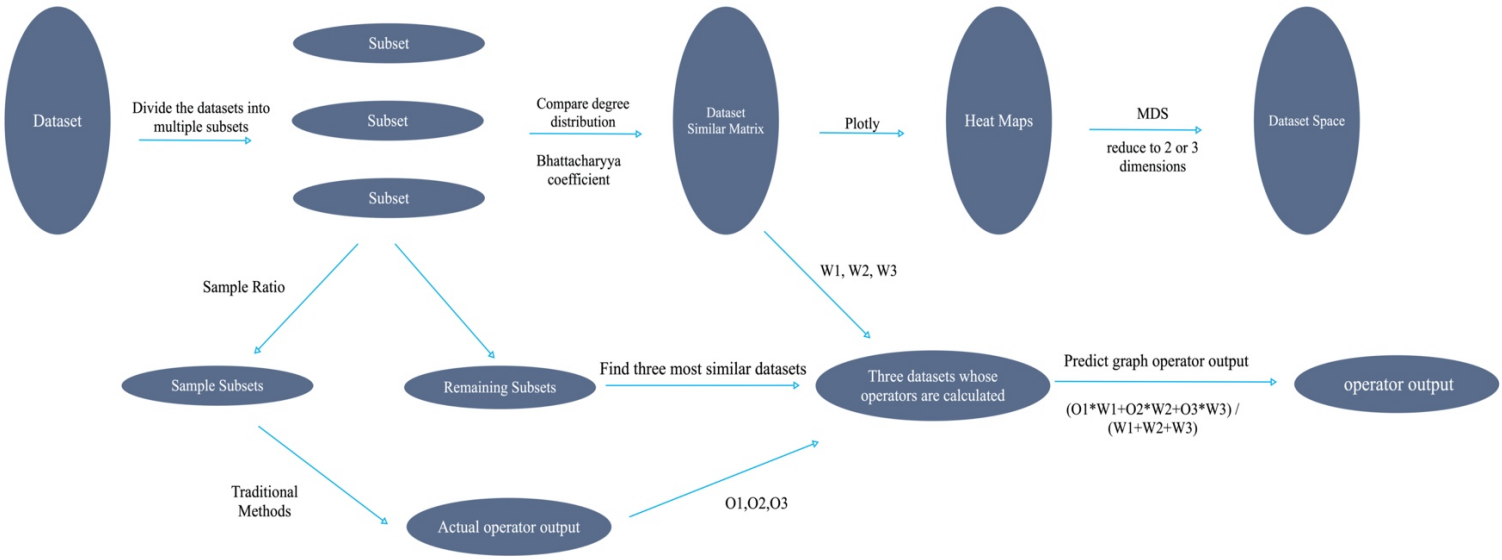

Figure 5 Apollo Framework

The implementation of temporal graph management system

To begin with, similarity matrix shall be calculated. However, it is quite difficult to compare the similarity of two graphs directly. Instead, graph degree distribution or other graph attributes could be used to express the similarity of graphs [3].

Currently, this system supports three similarity measures: Bhattacharyya coefficient, Jensen–Shannon divergence and Vertex count. Bhattacharyya coefficient and Jensen–Shannon divergence are two different algorithms to compare the degree distribution of graphs. Both of these two algorithms have the same range from 0 to 1. However, their output is completely opposite. Specifically, when two degree distributions are the same, the output of BC coefficient is 1. Instead, the output of JS divergence is 0. To resolve this problem, when the system uses JS divergence to compare the similarity of graphs, the output will change to 1 minus the value of JS divergence.

The system first divided the dataset into multiple sub datasets by months. Next, it calculated degree distribution of these graphs by Python. Then it applied JS divergence or BC coefficient to compare the degree distribution and generate the similarity matrix.

Apart from degree distribution, graph size could also be used to compare similarities between graphs.

$$s(G_i, G_j) = \frac{min(|V_{G_i}|,|V_{G_j}|)}{max(|V_{G_i}|,|V_{G_j}|)}. \quad (4)$$

Apparently, when two graphs have similar vertex counts, s would be approximate to 1, which means that these two graphs are very similar.

In addition, users are also able to combine similarity measures. For example, the system could compare two graphs based on their degree distribution and their graph size. Some simple linear composition formula could be adopted, like R = w1R1 + w2R2, with R1, R2 the degree distribution and vertex count similarity matrices respectively, and w1 = w 2 = 0.5.

With calculated similarity matrix, the system could visualize the output and generate corresponding heatmap and dataset space [5]. It first visualized the similarity matrix as a heatmap by Plotly. Next, it applied some dimension reduction algorithm to reduce the similarities matrix to three dimensions. The MDS (Multiple Dimensional Scaling) algorithm is a good option. MDS algorithm is an effective low-dimensional embedding algorithm. It could reduce high-dimensional data into low-dimensional space with the promise of ensuring that the

samples' distance between the original space and the low-dimensional space is consistent. Then, the system utilized Plotly's 3D scatter to build the dataset space construction.

Afterwards, based on the generated similarity matrix, the system could predict graph operator output, such as diameter, efficiently. To begin with, it selects some random data based on the given sample ratio. For these sample data, it calculates the value of graph operator using traditional methods. As for the rest of graphs, the system first finds three most similar datasets whose operators were calculated. Then it applies a weighted k-Nearest-Neighbours algorithm to predict graph operators for the rest of the graphs. However, this method also has limitation. At least three sample data are required. This may cause a problem for small datasets. For example, some datasets only last for a few months. If the sample ratio is 0.2, then the system might sample only one dataset in the end, which is obviously meaningless. To overcome this issue, a while loop was added in the code to check whether three samples have been selected. If not, the algorithm will sample additional data to meet the requirement.

To optimize the performance, the prediction function is only available after the existence of the similarity matrix. In this way, users could utilize one similarity matrix to experiment multiple graph operator predictions. For instance, users could predict diameter first, and then predict average shortest path without calculating the similarity matrix again.

3.2.4 Total Community Graph

To draw the graph of the number of total communities in different times, the system first divides the dataset by month. Then for each sub dataset, it uses community detection algorithm (Louvain algorithm) to extract the communities from the network. Then it utilizes Plotly to draw the total community graph.

Table 5 presents the data access interface of visualization components.

Table 5: data access interface of visualization components

Function of the interface	Interface
Generate data distribution map	def generateImg(df,type,fn,fr,to)
Calculate the degree distribution of dataset	def degree_distribution(data)
Create similarity matrix by js divergence	def js_distance(p, q)
Create similarity matrix by bc coefficient	def bhattacharyya(p, q)
Create similarity matrix by vertex count	def vertexcount(df,times)
Combine multiple similarity matrix	def combine_similarity(df,times,m1,m2,r)

The implementation of temporal graph management system

Generate heatmap and dataset space	def apollo(unixts,df,measure,m1=None,m2=None,r=None)
Generate graph operator diagram	def model(df,p,months,s,operator)
Change file to json format	def dftojson(old_df,duration,communities)
Generate total community graph	def caltotal(fn,df,months)

3.3 Community Structure

To explore the properties of the community structure, two research directions have been developed. One is community detection, whose purpose is to identify highly connected groups of objects within networks. The other one is called community search, which aims to extract high-quality communities based on some query nodes.

The goal of community detection is to identify a collection of nodes in a given network based on the network topology information. The obtained communities may overlap each other or may have a hierarchical relationship.

3.3.1 Implementation of local community detection

In terms of local community detection methods, they concentrate on the concepts of subgroup cohesiveness and mutuality, like k-cliques and k-cores. In this project, the system utilizes OLCPM [7], Online Label propagation CPM [6], to implement local detection method. This algorithm uses k-clique to define local community.

Figure 6 shows the procedure of OLCPM. The algorithm first implements OCPM, Online Clique Percolation Method, which takes two inputs: networks modification events and the clique size K. It then returns all k-clique communities. However, there are plenty of peripheral nodes are ignored by the first step. To address this problem, the second step, label propagation, is introduced. Based on the output communities of OCPM, it aims to discover the peripheral nodes and integrate them into the existing communities. For each community, it spreads an identity label and a distance weight to all the neighbouring peripheral nodes. When all labels have been shared, nodes belong to the communities with which they have the shortest distance. If the distances between the node and communities are the same, then the node belong to all of the communities. This project utilizes NetworkX, a Python library for studying graphs and networks, to implement this algorithm.

The implementation of temporal graph management system

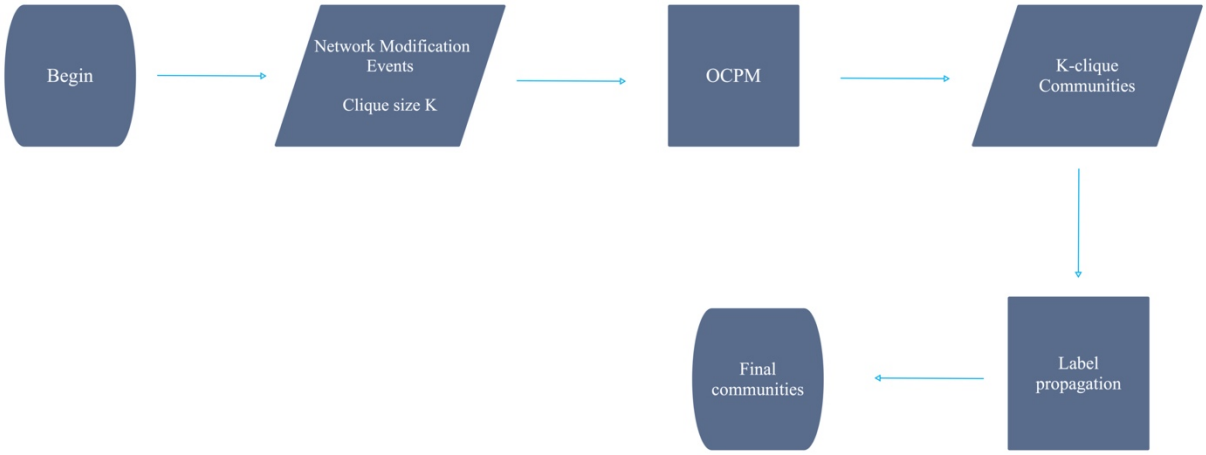

Figure 6 The procedure of OLCPM

3.3.2 Implementation of global community detection

In terms of the global communities, they are often seen as a property of the whole network. One popular global community definition relies on the widely used concept of modularity. This project uses Louvain algorithm [8] to implement global community detection.

Louvain algorithm is a method that could detect communities from large networks. It generally has two steps: At the beginning, each node is regarded as a community, and the weight of edges in the community is 0. For each node, the algorithm traverses all the neighbour nodes and measures the modularity gain by adding the node to their communities. It then places the node into the community that resulted in the greatest modularity increase. This process is repeated until the communities do not change. In the second step, it builds a new network by grouping all nodes of the same community. Then it calculates the weight of the edges between these newly generated communities and between all nodes within the communities. Then it goes to the step one for the next round. This project utilizes python-louvain package to implement Louvain algorithm.

3.3.3 Implementation of community search

Community search aims to identify communities that contain given query nodes. It is quite helpful to quickly obtain personalized community information. Unlike community detection, community search aims to find out high-quality communities based on some query nodes. To be more specific, given a query node v, the purpose of community search is to extract a community, which contains v and satisfies the properties of connectivity and cohesiveness. Multiple metrics, including k-core, k-truss, k-clique and k-ECC, have been introduced to

measure the cohesiveness of communities [9]. In this project, community search algorithm is designed to analyse the communities that a certain vertex belongs to in different periods. The system employs a local K-Core-Based community search algorithm for this project [10].

Figure 7 gives us an overall picture of the local K-Core-Based community search algorithm.

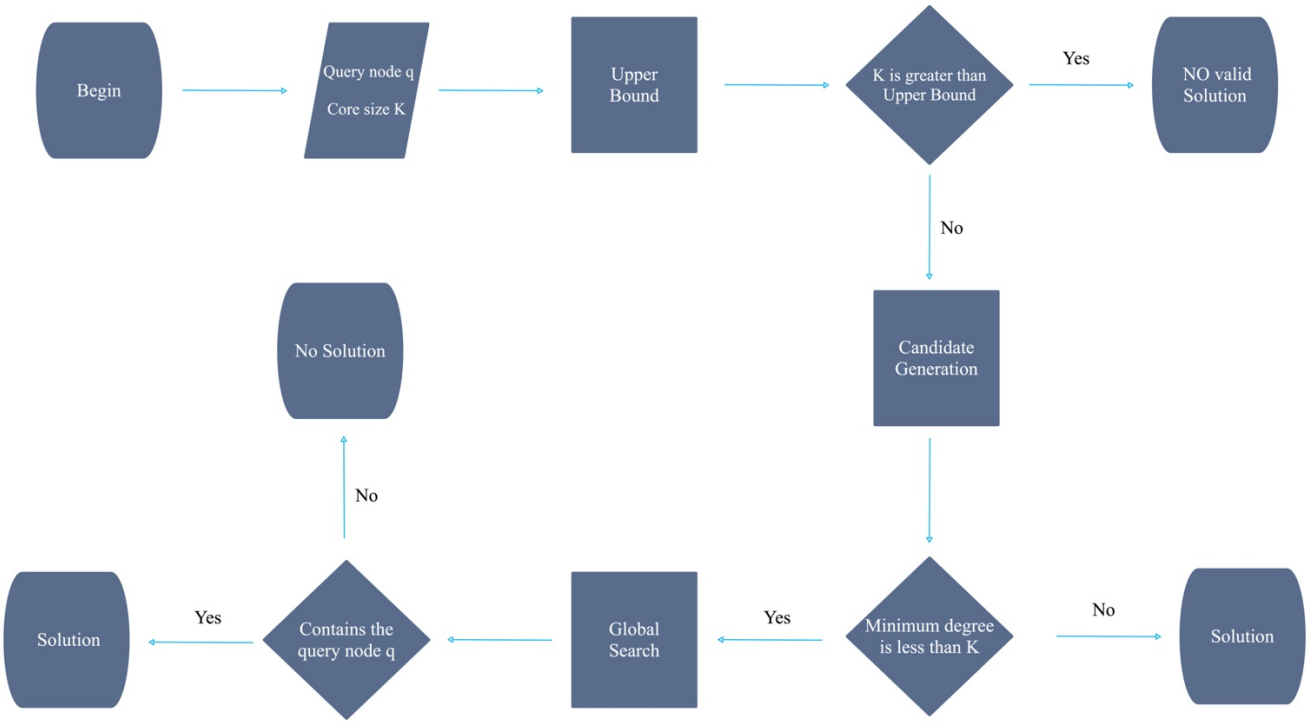

Figure 7 The flow chart of Community Search

Local K-Core-Based community search has three major steps. At first, it determines whether the graph G contains the solution of the community search given the vertex q. If core size k is larger than the "upper bound", then we can draw a conclusion that no valid solution could be found.

$$K \leq \lfloor \frac{1+\sqrt{9+8(|E|-|V|)}}{2} \tag{5}$$

Afterwards, the algorithm explores from the vicinity of the query node, and produces a candidate set C, which might involve a resolution to the problem. To optimize the performance of candidate selection, we choose the vertex with the largest number of connections to the selected vertices. However, sometimes, local information is insufficient to build a valid solution. To address this issue, a global search is performed on the k-cores of G [C]. This project utilizes NetworkX to implement this algorithm.

The implementation of temporal graph management system

Table 6 explains the data access interface of community structure.

Table 6: data access interface of community structure

Function of the interface	Interface
Extract local communities from the network	def olcpm(df,K)
Return k-clique communities from the graph	def ocpm(df,K)
Find the shortest distance between node and community	def findmini(num,ends,graph,other)
Extract a local community from the network with the query node v	def cst(df,duration,v,k)
Judge whether there is a solution for community search	def upperBound(G)
Generate a candidate C which may contain the solution	def candidateGeneration(graph, v, k)
Perform a global search on k-cores of G[C]	def globalsearch(graph,v,k)
Extract global communities from the network	def louvain(df)

Chapter 4: Results and Discussion

4.1 The test and discussion of the visualization components

Dataset for testing visualization components is soc-sign-bitcoin-alpha. This dataset is a temporal network with time interval between 2010 to 2016.

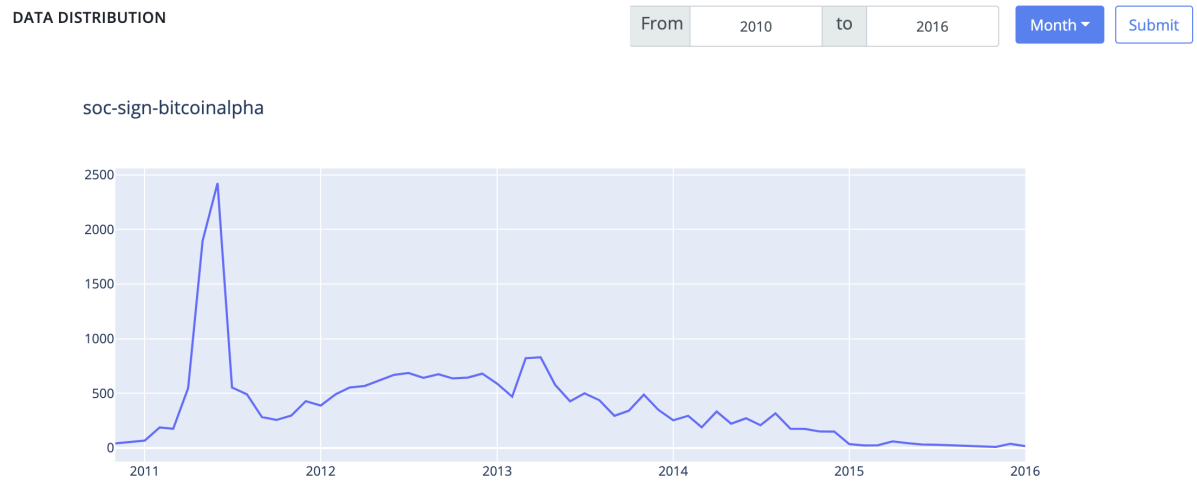

Figure 8 Data Distribution Map

Figure 8 is the data distribution map of soc-sign-bitcoin-alpha with the granularity of month. For each specific dataset, users could choose different time intervals and time granularities, including month, quarter and year. Then the system will generate the corresponding data distribution map.

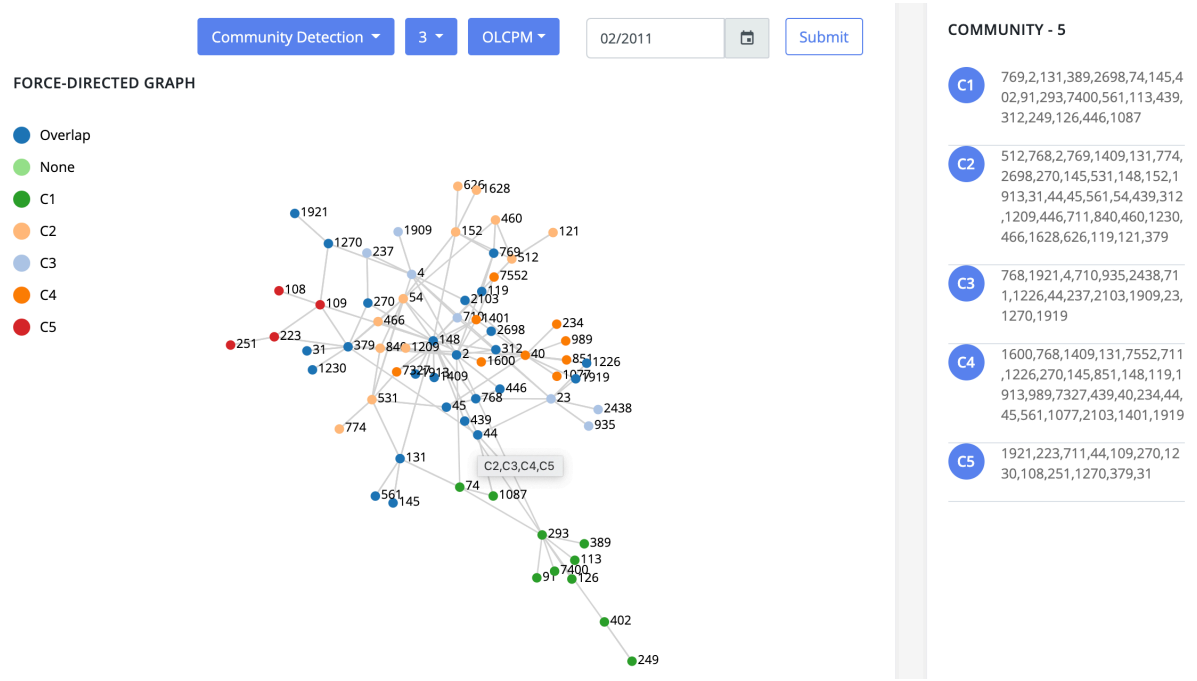

Figure 9 Force-directed Graph

Figure 9 is the force-directed graph, which depicts the network architecture of dataset soc-sign-bitcoinalpha in February 2011. In this graph, after running OLCPM community detection method (k=3), five communities have been found. Nodes whose community is one of them will have the corresponding colour. For node 44, whose legend is “Overlap”, if the mouse hovers onto this node, the graph displays C2, C3, C4 and C5. Therefore, node 44 belongs to all these communities at the same time.

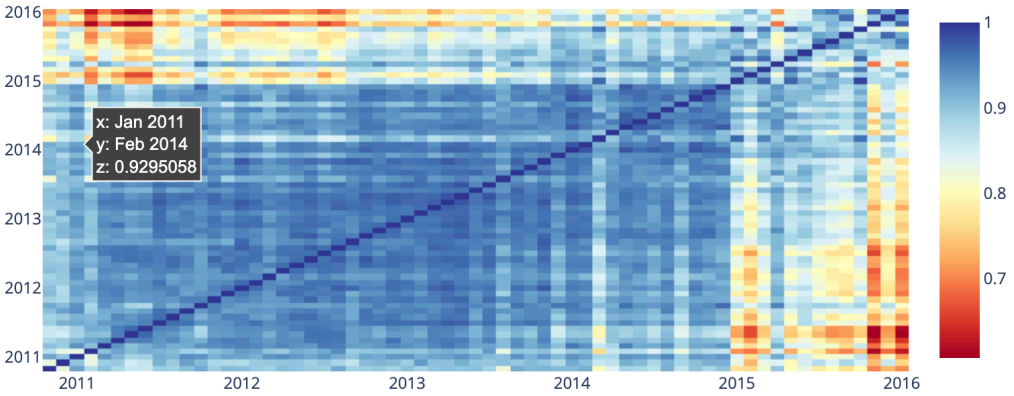

Figure 10 Heatmap

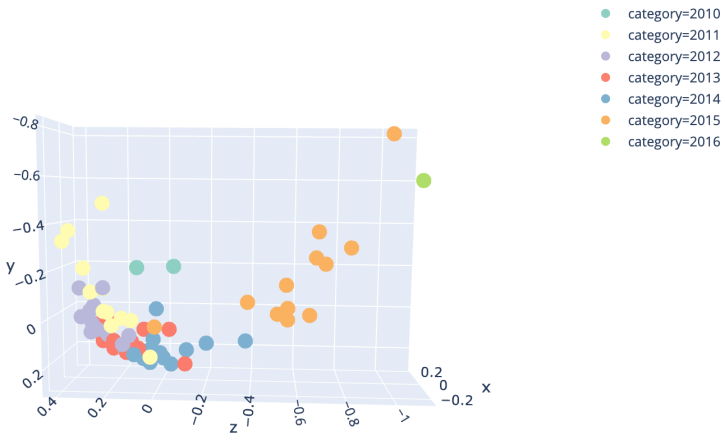

Figure 11 Dataset Space

Heatmap and datasets space could be used to display the similarities between datasets. Figure 10 is the heatmap of soc-sign-bitcoinalpha dataset. The similarity matrix was generated by utilizing Bhattacharyya Coefficient method. In Figure 10, it is obvious that the bluer the colour, the more similar two graphs are; the redder the colour, the less similar two graphs are.

To give users more direct sense of the similarities between datasets, a dataset space was created. As displayed in the figure 11. Each subset was projected into a point in this three-dimension space. The closer the points, the more similar these datasets are. The system also adopted different colours for the points based on their time period (year), which could help users

The implementation of temporal graph management system

differentiate these points rapidly.

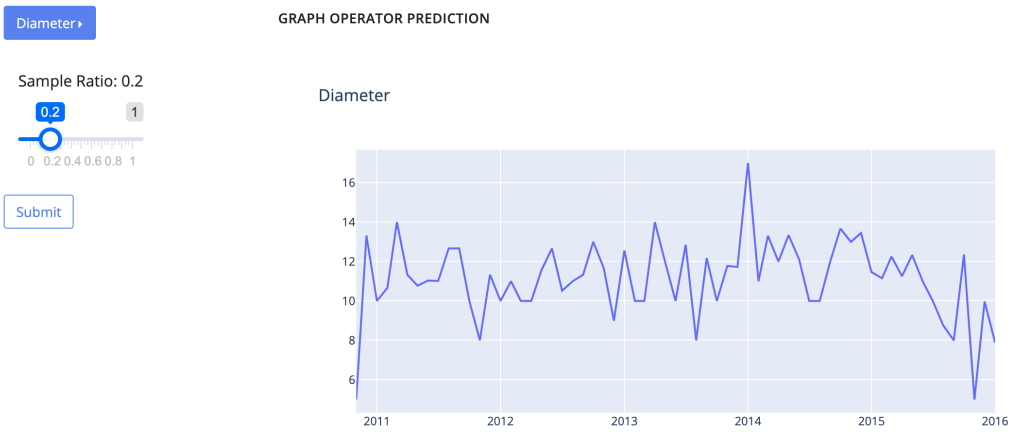

Figure 12 Diameter Prediction Diagram

Figure 12 shows the diameter prediction diagram of dataset soc-sign-bitcoinalpha. While predicting graph operator outputs, users should first select the graph operator they want to predict. Next, they need to choose the sample ratio of the dataset. When the sample ratio is small, the system could create graph operator diagrams very quickly. However, the accuracy of the graph operator output may be compromised.

4.2 The test and discussion of the community structure

Dataset for testing community structure is sx-mathoverflow-a2q. This dataset is a temporal network with time span between 2009 to 2016.

When drawing the force-directed graph, users could choose community detection or community search methods, then the system will display the calculated communities on the right side.

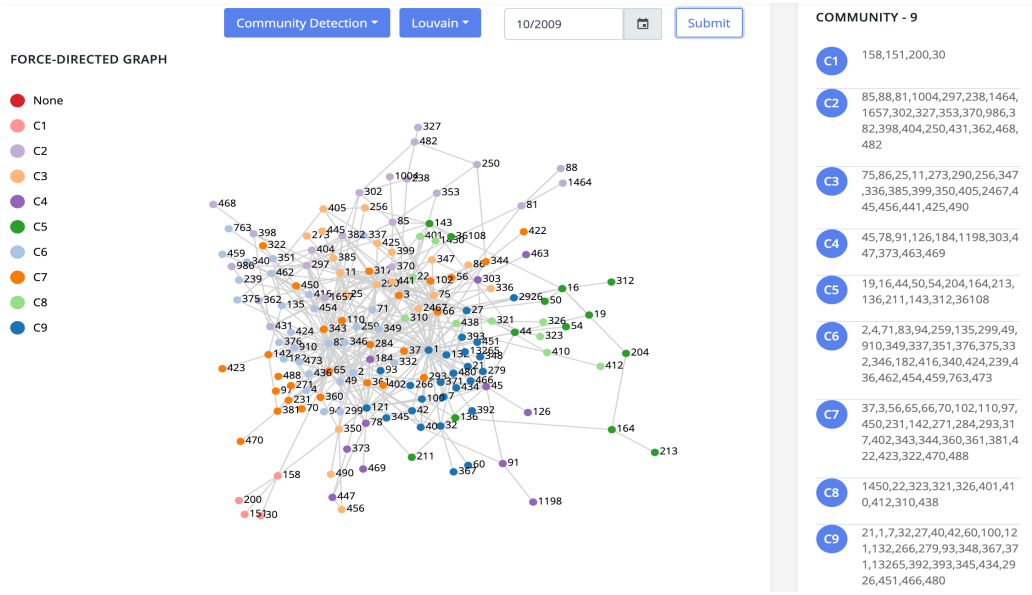

Figure 13 Community detection force-directed graph

The implementation of temporal graph management system

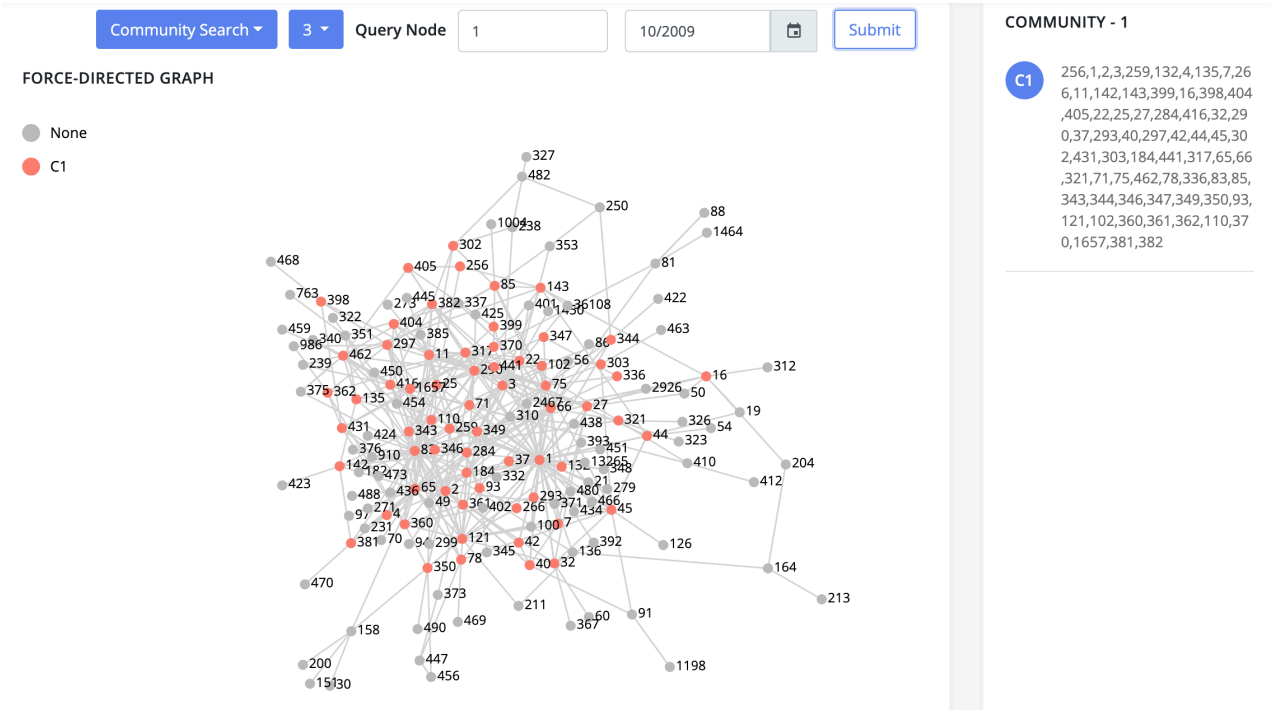

Figure 14 Community Search Force-directed Graph

Figure 13 and figure 14 are the force-directed graphs of dataset sx-mathoverflow-a2q in October 2009. As is shown by these two graphs, calculated communities are displayed on the right side of the graphs.

Communities in figure 13 is calculated by Louvain algorithm. From figure 13, users could effortlessly ascertain which community the nodes belong to. With the legend, it is easy to associate the nodes in the graphs with the communities displayed on the right side.

Communities in figure 14 is found by K-cored-based community search method with the query node 1, and the k-core size 3. Since this method could only extract one maximal community from the network, the system assigned red colour to nodes within the community. For background nodes, the system used grey colour to represent them.

4.3 The test and discussion of the whole system

In this part, several tests were carried out to demonstrate the efficiency and effectiveness of the temporal graph management system.

Datasets: The datasets for testing the whole system involve CollegeMsg, email-Eu-core-temporal, soc-sign-bitcoin-alpha, soc-sign-bitcoin-otc and sx-mathoverflow. All these temporal datasets are from SNAP.

The implementation of temporal graph management system

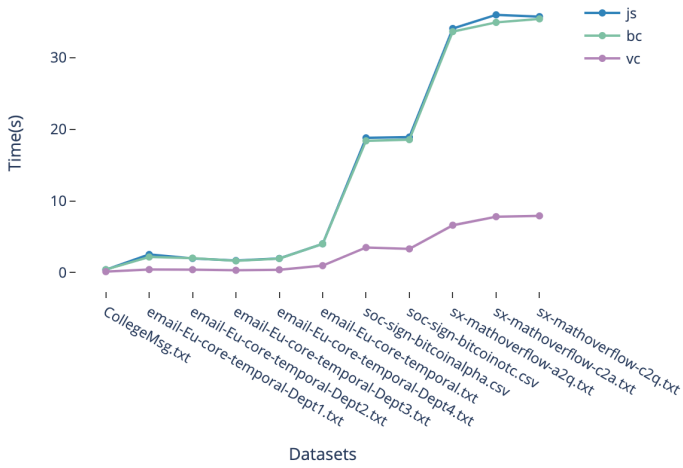

Figure 15 Similarity matrix time cost

Figure 15 displays the time consumption for creating similarity matrix by three different algorithms. As shown in Figure 15, this system could generate the similarity matrix efficiently with vertex count method. Additionally, if users want to create similarity matrix by comparing the degree distribution of graphs, both JS divergence and BC coefficient are equally efficient to provide this service.

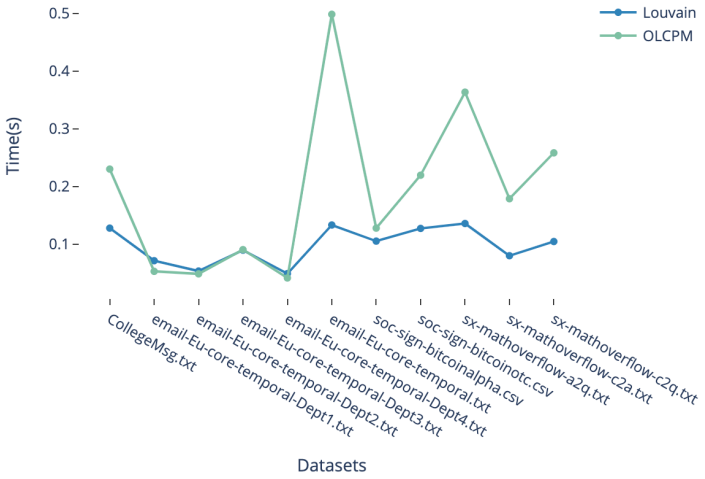

Figure 16 Community detection time consumption

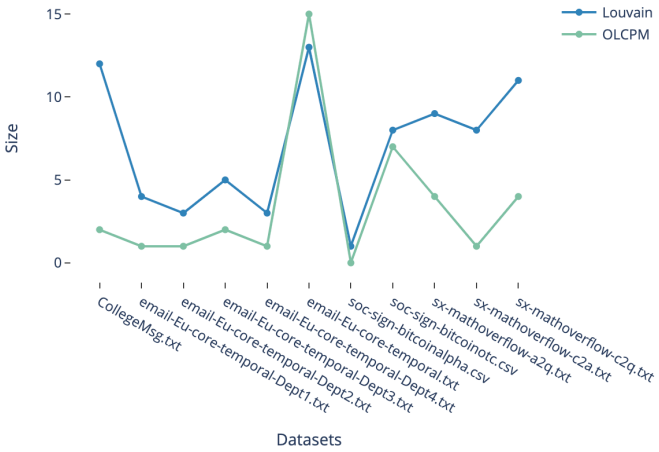

Figure 17 the Number of communities

Figure 16 depicts the time consumption of two community detection methods: OLCPM and Louvain. Figure 17 shows the number of communities that are extracted from the network by these two algorithms. As displayed in figure 16 and figure 17, both local and global community detection methods in this system could extract communities from the network efficiently. With Louvain algorithm, the system could detect plenty of communities in a short time. OLCPM, instead, enables the system to detect larger and more cohesive communities efficiently.

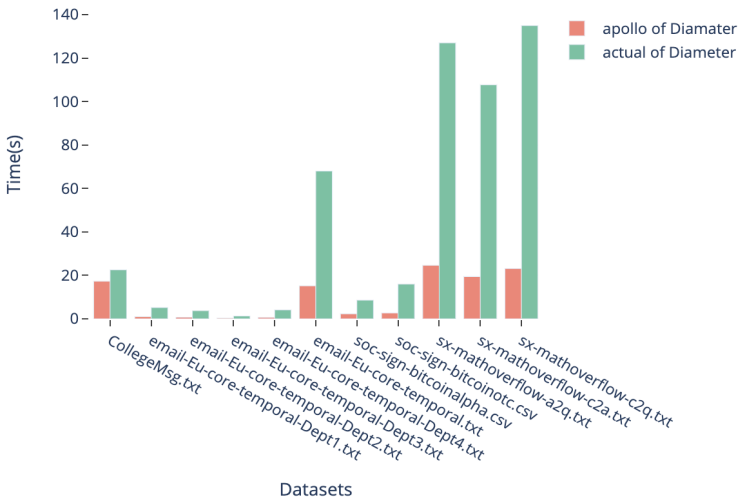

Figure 18 Graph operator output time cost

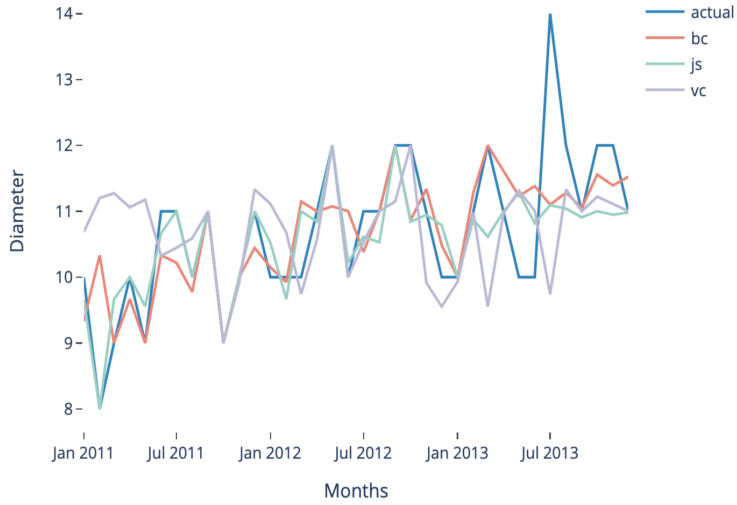

Figure 19 Graph operator output value

Figure 18 displays the time spent for obtaining the diameter for all the datasets. The similarity matrix was generated by BC coefficient method. From figure 18, it is obvious that this system could predict graph operator outputs much more efficiently than traditional method, especially for large datasets.

Figure 19 depicts the diameter value of dataset sx-mathoverflow from 2011 to 2013. The actual value of diameter is calculated by traditional method, while the other three are predicted with different similarity matrices by the system. As depicted in figure 19, all these three similarity matrices are powerful enough to predict graph operator output accurately with a small bias.

Chapter 5: Conclusion and Further Work

5.1 Conclusion

In this project, a temporal graph management system was designed to help users have a deep understanding of the temporal networks. This system provides a great deal of useful information of temporal network, including metadata and statistical data. Besides, assorted visualization components have been created so that users could easily analyse the properties of temporal graphs, such as data distribution, network architecture and dataset similarity. Based on Apollo framework, this system is capable of predicting graph operator outputs efficiently and accurately. What's more, this project also enables users to mine the patterns of community structure through community detection methods and community search algorithm. This system supports multiple user-friendly services as well. For example, users could upload their own datasets to acquire a detailed network analysis of their temporal datasets. Furthermore, a specification document is also provided in this system so that users could gain a better understanding of the project and operate the system functions correctly.

5.2 Further Work

To begin with, until now, this project only focuses on the temporal attribute of network and neglects other important attributes. For example, for directed graphs, this system may fail to extract some useful information, since our system treats all the networks as undirected. Therefore, supporting other types of networks is crucial for this system.

Secondly, some functions in this system still have space for improvement. Since not all the functions in the system have direct well-written code, the system builds these complicated functions, like OLCPM and community search algorithm, imperfectly. As a result, the time complexity increases, which may affect users' experience. Hence, a more efficient and effective system should be developed.

Last but not the least, this project will support login and register functions. With these additional functions, users could save datasets into their collection, which is very convenient for users to navigate them afterwards. Besides, users may also want to upload some private datasets. For example, some datasets are very sensitive. By uploading these datasets publicly may infringe the rights or interests of other people. If users could have the option to decide whether or not to share their datasets to others, this kind of awful scenario could be avoided.

References

[1] Bhattacharyya, A. (1943). On a measure of divergence between two statistical populations defined by their probability distributions. *Bulletin of the Calcutta Mathematical Society*, 35(1), pp.99-109.

[2] Symeon, P., Yiannis, K., Athena, V. and Ploutarchos, S., (2012). Community detection in social media, performance and application considerations. *Journal of Data Mining Knowledge Discovery*, 24(3), pp.515-554.

[3] Bakogiannis, T., Giannakopoulos, I., Tsoumakos, D. and Koziris, N., (2018). *Graph Operator Modeling over Large Graph Datasets*. [ONLINE] Available at: https://arxiv.org/abs/1802.05536. [Accessed 31 March 2020].

[4] Bakogiannis, T., Giannakopoulos, I., Tsoumakos, D. and Koziris, N., (2019), June. Predicting Graph Operator Output over Multiple Graphs. *In International Conference on Web Engineering* (pp. 107-122). Springer, Cham.

[5] Bakogiannis, T., Giannakopoulos, I., Tsoumakos, D. and Koziris, N., (2019), June. Apollo: A Dataset Profiling and Operator Modeling System. *In Proceedings of the 2019 International Conference on Management of Data* (pp. 1869-1872).

[6] Palla, G., Derényi, I., Farkas, I. and Vicsek, T., (2005), Uncovering the overlapping community structure of complex networks in nature and society. *nature*, 435(7043), pp.814-818.

[7] Boudebza, S., Cazabet, R., Azouaou, F. and Nouali, O., (2018). OLCPM: An online framework for detecting overlapping communities in dynamic social networks. *Computer Communications*, 123, pp.36-51.

[8] Blondel, V.D., Guillaume, J.L., Lambiotte, R. and Lefebvre, E., (2008). Fast unfolding of communities in large networks. *Journal of statistical mechanics: theory and experiment*, 2008(10), pp.10008.

[9] Fang, Y., Huang, X., Qin, L., Zhang, Y., Zhang, W., Cheng, R. and Lin, X., (2020). A survey of community search over big graphs. *The VLDB Journal*, 29(1), pp.353-392.

[10] Cui, W., Xiao, Y., Wang, H. and Wang, W., (2014), June. Local search of communities in large graphs. *In Proceedings of the 2014 ACM SIGMOD international conference on Management of data* (pp. 991-1002).

Acknowledgement

First and foremost, I would like express my sincere gratitude and respect to my supervisor, who inspired and supported me during this whole process. He is an expert in the area of graph theory and data mining etc. What's more, he is also very patient and insightful. Whenever I encountered a problem, he would share some useful hints, which helped me address the issue. Besides, he always recommended fascinating lectures to us. Without his guidance, I do not believe I could finish my project in this way. In addition, I also received a lot of help from the students in lab. I am very grateful for their selfless dedication.

Secondly, I want to thank my family. They are always so supportive and considerate. The final year in university is very challenging. Thanks to my beloved parents and sister. It is their company that makes me get through this year successfully.

Finally, I have to show my appreciation to my friends. They play such an important role in my life. As I've stated before, the final year in school is so tough. Fortunately, I have met some best friends. We gave each other courage and confidence to confront the obstacle in our life.

Risk and environmental impact assessment

This project focuses on developing a temporal graph management system. Table 7 shows us the risk assessment of this project. The risk analysis first describes the risk and corresponding impact. Afterwards, the rating of likelihood and impact are estimated. Last, the preventative actions for reducing the risk are discussed.

Table 7: Risk Assessment Table

Description of Risk	Description of Impact	Likelihood rating	Impact rating	Preventative actions
Loss or damage of the system's source code	The system could not operate normally	3	5	Backup the source code
Hackers carry out DDOS attack to the server	The system cannot process the request from users	2	4	Block access to abnormal traffic; Only open necessary ports in the server
Loss or damage of the datasets/database data	The whole system could not perform normally	2	4	Backup the files and data in database.

Cost of manufacture: As a software implementation project, this option is not applicable.

Waste disposal and recycling: Since this project is a web-based system, no hardware is required. Thus, there is no issue of waste disposal and recycling.

Energy use in service and Savings in energy: This system is a software implementation project, which contributes to energy saving and environment protection.

Appendix

北京邮电大学 本科毕业设计（论文）任务书

Project Specification Form

Part 1 – Supervisor

论文题目 **Project Title**	The implementation of temporal graph management system		
题目分类 **Scope**	Data Science and Artificial Intelligence	Implementation	Software
主要内容 **Project description**	A temporal graph, also known as a time-varying graph, is a graph whose links are active only at certain points in time. The exploration of such temporal graphs is a fundamental problem for graph analysis. Currently, there are a few systems that simply display the basic statistics of dynamic temporal graphs. A prominent example is SNAP (Stanford Network Analysis Project), a highly famous website in this field. However, it does not provide user-friendly services for users to further explore and understand the properties of the graph. To resolve this issue, we plan to build a temporal graph management system to show various information about the dynamic temporal network, including both basic statistics and deep insights. With the help of this systems, users can easily explore the properties of graphs, and mine structural patterns (e.g., community) on top of the temporal graph. At the aspect of structural mining, we will use community detection methods such as OLCPM to extract the community structure. Furthermore, we may also design some community search algorithms to analyze the communities that a certain vertex belongs to in different periods. In addition, the system will be designed as a website so that user can easily access the system through internet.		
关键词 **Keywords**	Graph Analysis, Database, Data Management		
主要任务 **Main tasks**	**1** Design the system architecture from data storage to the logic of management. Use Django framework to build a website for displaying the information about dynamic temporal network.		
	2 Design visualization component so that users can easily explore the dynamic graph;		
	3 Implement efficient community detection methods to extract the dynamic community structure of an evolving network. The detection method will support both global and local community detection.		
	4 Test the efficiency of the system and develop corresponding optimization techniques.		
主要成果 **Measurable outcomes**	**1** A web-based dynamic graph management systems. People can use the system to view the basic information of the temporal network such as nodes, temporal edges, time span, and the snapshot of dynamic graphs.		
	2 A specification document that describes the function of the system in details.		
	3 The system equips with a graph mining component, and it supports community detection at least.		

北京邮电大学 本科毕业设计（论文）任务书

Project Specification Form

Part 2 - Student

<table>
<tr><td>学院
School</td><td>International School</td><td>专业
Programme</td><td colspan="3">e-Commerce Engineering with Law</td></tr>
<tr><td>姓
Family name</td><td>Yu</td><td>名
First Name</td><td colspan="3">Weiqiang</td></tr>
<tr><td>BUPT 学号
BUPT number</td><td>2016213346</td><td>QM 学号
QM number</td><td>161196754</td><td>班级
Class</td><td>2016215113</td></tr>
<tr><td>论文题目
Project Title</td><td colspan="5">The implementation of temporal graph management system</td></tr>
<tr><td>论文概述
Project outline

Write about 500-800 words

Please refer to Project Student Handbook section 3.2</td><td colspan="5">1.An initial analysis of user requirements, and how data will be collected:

A temporal graph, also known as a time-varying graph, is a graph whose links are active only at certain points in time. The exploration of such temporal graphs is a fundamental problem for graph analysis. Currently, there are a few systems that simply display the basic statistics of dynamic temporal graphs. A prominent example is SNAP (Stanford Network Analysis Project), a highly famous website in this field. However, it does not provide user-friendly services for users to further explore and understand the properties of the graph. To resolve this issue, we plan to build a temporal graph management system to show various information about the dynamic temporal network, including both basic statistics and deep insights. With the help of this systems, users can easily explore the properties of graphs, and mine structural patterns (e.g., community) on top of the temporal graph. At the aspect of structural mining, we will use community detection methods such as OLCPM to extract the community structure. Furthermore, we may also design some community search algorithms to analyse the communities that a certain vertex belongs to in different periods. In addition, the system will be designed as a website so that user can easily access the system through internet.

For the way data is collected, we choose to gather the datasets from SNAP (Stanford Network Analysis Project), which is a general purpose network analysis and graph mining library.

2.The algorithms, methodologies and other techniques to be employed

We use Apollo framework to predict graph operator output over multiple graphs. Besides, we adopt community detection algorithms, such as OLCPM, to extract the dynamic community structure of an evolving network. Furthermore, we may use some community search algorithms, such as K-Core-Based Community Search, to analyse the communities that a certain vertex belongs to in different periods.

3.An initial specification of how users will interact with the system (implementation)
Users are capable of interacting with the system by the following two approaches:
1. Visualization Components:</td></tr>
</table>

Users first select a dataset, a time interval and time granularity, such as month, quarter and year. The system will then create the corresponding data distribution map and force-directed graph. Furthermore, by choosing a dataset and a time interval, the system will generate the corresponding heatmaps and dataset space, which reveals the similarities of this dataset among different months.

2. Community Structure:
   Users first select a dataset, a k value (the size of the clique) and a time interval. Then our system will list all the communities of that certain time span. Besides, Users could select a specific point, and then our system will output all communities this certain vertex belongs to in different times.

4.Programming language / database/ software package and hardware to be used

Language: Python 3.7

Database: SQLite3 database.

Software Package: Django

Hardware: Linux ubuntu

5.A list of background material consulted including World Wide Web pages

[1] Souâad Boudebza, Rémy Cazabet, Faiçal Azouaou, Omar Nouali (2018). OLCPM: An Online Framework for Detecting Overlapping Communities in Dynamic Social Networks. arXiv.org. Available at: https://arxiv.org/abs/1804.03842

[2] Tasos Bakogiannis, Ioannis Giannakopoulos, Dimitrios Tsoumakos, and Nectarios Koziris. (2019). Apollo: A Dataset Profiling and Operator Modeling System. ACM. Available at: https://doi.org/10.1145/3299869.3320220

[3] Tasos Bakogiannis, Ioannis Giannakopoulos, Dimitrios Tsoumakos, Nectarios Koziris. (2019) Predicting Graph Operator Output over Multiple Graphs. ICWE Available at: https://link.springer.com/chapter/10.1007/978-3-030-19274-7_9

[4] Tasos Bakogiannis, Ioannis Giannakopoulos, Dimitrios Tsoumakos, Nectarios Koziris (2018). Graph Operator Modeling over Large Graph Datasets. arXiv.org. Available at: https://arxiv.org/abs/1802.05536

[5] Martin Rosvall, Jean-Charles Delvenne, Michael T. Schaub, Renaud Lambiotte. (2017). Different approaches to community detection. arXiv.org. Available at: https://arxiv.org/abs/1712.06468 .

<table>
<tr><td></td><td>[6] Oualid Boutemine and Mohamed Bouguessa. (2017). MCDA: A Parameterless Algorithm for Detecting Communities in Multidimensional Networks. ACM, Available at:https://doi.org/10.1145/3110025.3110052

[7] Twan Van Laarhoven and Elena Marchiori. (2016). Local network community detection with continuous optimization of conductance and weighted kernel K-means. ACM, Available at: https://dl.acm.org/citation.cfm?id=3007100

[8] Renzo Angles, Claudio Gutierrez (2017). An introduction to Graph Data Management. arXiv.org. Available at: https://arxiv.org/abs/1801.00036.

[9] Yixiang Fang, Reynold Cheng, Siqiang Luo, and Jiafeng Hu. (2016). Effective community search for large attributed graphs. ACM, Available at: https://doi.org/10.14778/2994509.2994538

[10] Yixiang Fang, Xin Huang, Lu Qin, Ying Zhang, Wenjie Zhang, Reynold Cheng, Xuemin Lin. (2019). A Survey of Community Search over Big Graphs. arXiv.org. Available at: https://arxiv.org/abs/1904.12539.</td></tr>
<tr><td rowspan="2">道德规范
Ethics</td><td>Please confirm that you have discussed ethical issues with your Supervisor using the ethics checklist (Project Handbook Appendix 2).
[YES/NO]</td></tr>
<tr><td>Not applicable</td></tr>
</table>

中期目标 **Mid-term target.** **It must be tangible outcomes, E.g. software, hardware or simulation.** **It will be assessed at the mid-term oral.**	Target1: A web-based dynamic graph management systems. Target2: Design a variety of visualization components in our system. Target3: Implement local community detection method.

Work Plan (Gantt Chart)

Fill in the sub-tasks and insert a letter X in the cells to show the extent of each task

	Nov 1-15	Nov 16-30	Dec 1-15	Dec 16-31	Jan 1-15	Jan 16-31	Feb 1-15	Feb 16-29	Mar 1-15	Mar 16-31	Apr 1-15	Apr 16-30
Task 1 **Design the system architecture from data storage to the logic of management.**												
Sub-Task1: Collect the datasets from SNAP library	X											
Sub-Task2: Build a website in Django and store the data into SQLite3 database.		X	X	X								
Sub-Task3: Use Apache to deploy the project on Linux					X	X						
Task 2 **Design visualization component so that users can easily explore the dynamic graph**												
Sub-Task1: Create data distribution map	X	X										
Sub-Task2: Design the force-directed graph		X	X									
Sub-Task3: Generate the heatmap and datasets space according to Apollo algorithms			X	X								
Sub-Task4: Draw the graph of the number of total communities in different times					X	X						
Task 3 **Implement efficient community detection methods to extract the dynamic community structure of an evolving network**												
Sub-Task1: Implement local community detection method				X	X							
Sub-Task2: Implement global community detection method					X	X						
Sub-Task3: Implement community search algorithm							X	X	X			
Task 4 **Test the efficiency of the system and develop corresponding optimization techniques.**												
Sub-Task1: Test software and system									X	X		
Sub-Task2: Optimization of system										X	X	
Sub-Task3: Complete the report of the system									X	X	X	X

北京邮电大学 本科毕业设计（论文）初期进度报告

Project Early-term Progress Report

学院 School	International School	**专业 Programme**	**e-Commerce Engineering with Law**		
姓 Family name	Yu	**名 First Name**	Weiqiang		
BUPT 学号 BUPT number	2016213346	**QM 学号 QM number**	161196754	**班级 Class**	2016215113
论文题目 Project Title	The implementation of temporal graph management system				

已完成工作 Finished work:

1. Collect the datasets from SNAP library

Stanford Network Analysis Platform (SNAP) is a general purpose network analysis and graph mining library. It contains a great many of large network datasets, including temporal networks. SNAP not only provides users the origin data file but also supplies some dataset statistics information. For my final project, I downloaded several datasets from SNAP, such as sx-stackoverflow, sx-mathoverflow and email-Eu-core-temporal. To store these datasets, I decide to design two database schemas.

Table Name	Description
Datainfo(Id, Name, Type, Nodes, Temporal Edges, Static Edges, Description)	Contains the statistical information of temporal network datasets. **Id**: the id of the network **Name**: the name of the network **Type**: The type of the network, like directed, weighted, temporal. **Nodes**: Number of nodes in the network **Temporal Edges**: Number of temporal edges **Static Edges**: Number of static edges **Description**: Other details of the network
Subdata(Id, Name, TimeSpan, Nodes, Temporal edges, static edges)	Stores the statistical information of specific dataset. **Id**: the id of the dataset **Name**: the name of the dataset **TimeSpan**: time period of the dataset in days **Nodes**: Number of nodes in the dataset **Temporal Edges**: Number of temporal edges **Static Edges**: Number of static edges

Problem: How to process the data in downloaded files:
Solution: Use pd.read_csv() function to change it to DataFrame, for further data processing.

2. Build a website in Django and store the data into SQLite3 database.

I choose Django as my web framework to build the temporal graph management system. Django is a high-level Python Web framework that encourages rapid development and clean, pragmatic design. Besides, Django also supports SQLites3 as the database by default. Therefore, we do not need to configure the database. What's more, Django also has an automatic admin interface. It reads metadata from models to provide a quick, model-centric interface where trusted users can manage content on your site. Thus, it is very convenient for website managers to interact with the database.

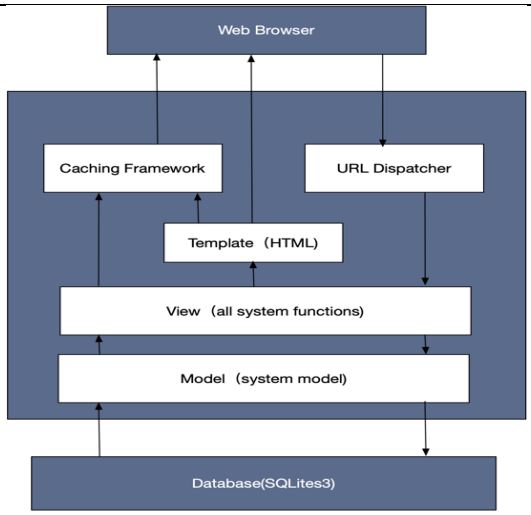

Figure 1: Django Framework

Problem: How to use Django to build a website?
Solution: Could read official documents and watch some instructional videos.

3. Create data distribution map

In order to let users intuitively see the distribution of data in different time span, I created some data distribution map. I first used Pandas to divide the dataset based on the given time granularity, such as year, month. Afterwards, I calculated the sum of the amount of data in different time. In the end, I adopted Plotly, which is an online graphing, analytics, and statistics tools, to draw the distribution map based on the calculated result.
Problem: Given Unix timestamp, how to convert it to date?
Solution: Use some Python package, e.g. datetime.datetime.fromtimestamp(timestamp).strftime('%Y-%m-%d %H:%M:%S').

4. Design the force-directed graph

A force-directed diagram often exposes the inherent symmetric and clustered structure of a graph and shows a well-balanced distribution of nodes with few edge crossings. To design the force-directed graph, I first used Python to convert the original data into JSON files and then utilized D3.js, which is a JavaScript library for visualizing data using web standards, to draw the graph.

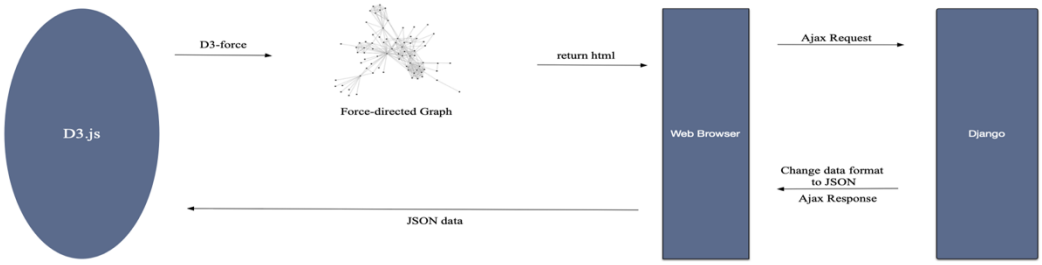

Figure 2: Force-directed Graph

Problem: How to use D3.js to draw the force-directed graph?
Solution: Could apply D3's force layout which uses a physics based simulator for positioning visual elements.

5. Generate the heatmap and datasets space according to Apollo algorithms

Apollo is based on the observation that datasets with similar statistical properties affect a wealth of real world operators in similar ways and, hence, lead them to produce similar

outputs. The similarity matrix can be visualized as a heat map, the dataset space can be reduced to 2 or 3 dimensions and be displayed in an interactive chart. First, I adopted Bhattacharyya coefficient to calculate the similarities of degree distribution among datasets and generate the heatmap. Next, I applied multidimensional scaling (MDS) to reduce the similarities matrix to three dimensions, and then used Plotly's 3D scatter to build the dataset space construction.

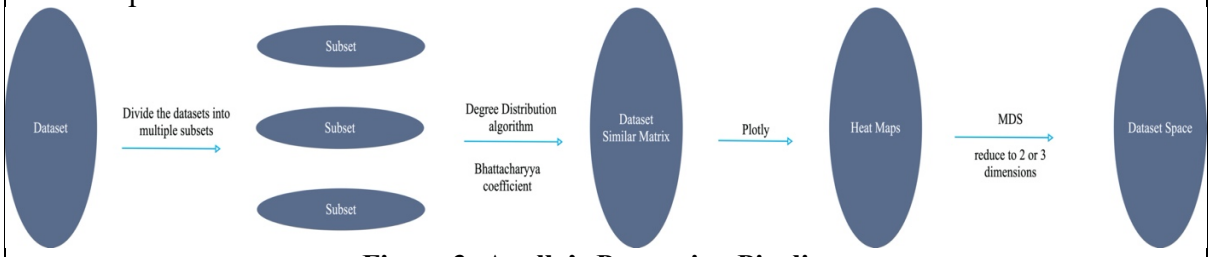

Figure 3: Apollo's Processing Pipeline

To speed up the page load, I decided to store the heatmap and dataset space in database. I first calculated the similarity matrix in advance, then utilized Plotly to generate corresponding html and then stored it in the database. Now, the website can directly retrieve the heatmap and dataset space from the database without a huge computation effort.

The database schema is following:

Table Name	Description
Subdata (Id, Name, TimeSpan, Nodes, Temporal edges, static edges, **Heatmap**, **Dataspace**)	Append two attributes **Heatmap:** the heatmap of the dataset in different time **Dataspace:** the dataset space of the dataset in different time
Graphinfo (Id, Heatmap, Dataspace)	Contains the graph information of all datasets **Heatmap:** the heatmap of different datasets **Dataspace:** the dataset space of different datasets

Problem: How to compare the similarity between two graphs?

Solution: Use degree distribution of two graphs to express the similarity of graphs.

是否符合进度？On schedule as per GANTT chart?

[YES/NO]

YES

下一步 Next steps:

1. Use Apache to deploy the project on Linux

Until now, all my work only runs in my own environment. To allow more people to have access to the system. I choose to use Apache, a free and open-source cross-platform web server software, to deploy the project on Linux.

In the process, it is necessary to tackle the following issues:

1.How to deploy Django in Apache?

2.How to configure Apache in Linux environment?

2. Implement local and global community detection method

The field of community detection aims to identify highly connected groups of individuals or objects inside networks, these groups are called communities. To help users better understand the community structure, I plan to implement both local and global community detection methods.

In the process, it is necessary to tackle the following issues:

1.How to implement local community detection method in our system?

2.How to implement global community detection method in our system?

北京邮电大学 本科毕业设计（论文）中期进度报告

Project Mid-term Progress Report

学院 School	International School	**专业 Programme**	**e-Commerce Engineering with Law**		
姓 Family name	Yu	**名 First Name**	Weiqiang		
BUPT 学号 BUPT number	2016213346	**QM 学号 QM number**	161196754	**班级 Class**	2016215113
论文题目 Project Title	The implementation of temporal graph management system				

是否完成任务书中所定的中期目标？Targets met (as set in the Specification)?
[YES/NO]
YES

已完成工作 Finished work:

1. A web-based dynamic graph management systems

For our system, I designed multiple pages to display the dynamic temporal network information for users. For the index page, people could see the general information of the datasets, like network types and network statistics data. Besides, the index page also contains the heat map and dataset space, which indicate the similarities between the datasets. What's more, users could have access to the original data files by downloading them from the website. On the data presentation page, users could see the details of each specific dataset. Furthermore, this page also includes several visualization components, such as force-directed graph, so that users are able to explore and understand the properties of the graph easily. I also designed multiple error handling mechanisms. For example, if a user enters an invalid time interval, the website will remind the user that their input is incorrect. In order to let more people get access to the website, I deployed the project on Linux. I first installed Apache2 and Virtualenv, which is a tool to create isolated Python environments. Secondly, I needed to configure Apache2. I used libapache2-mod-wsgi-py3 as the adapter to provide a WSGI compliant interface for hosting Django within Apache. Afterwards, I modified the Apache's configuration file. Then, the website could be accessed successfully. However, the static files, such as css and JavaScript are not loading. To resolve this issue, I first created a place to store static files on server. Then I modified the settings file of Django. And now, the static files can be served as expected. The website IP address is 152.136.59.62 and the port is 8080.

2. Design a variety of visualization components in our system

Currently, I have designed multiple visualization components. Data distribution map is useful to display the distribution of the data in different time intervals. Users are capable to choose the time granularity they want, and the system will generate the corresponding graph. Force-directed graph helps users understand the network architecture intuitively. Users can choose local or global community detection method, then the website will display the calculated communities. In order to represent all the nodes in the graph with different colors based on their communities, I assign an extra attribute, community, to all the nodes. Then I used Python to change the original files into JSON format, and utilized D3js to draw the force-directed graph. By doing this, it is very convenient for people to recognize different communities in the graph. What's more, inspired by the framework of Apollo, I also created heatmap and dataset space to visualize the similarities between datasets. Since it is difficult to compare the similarity of two graphs, I used the degree distribution to express the similarity of graphs. Now, I have designed three similarity measure: Bhattacharyya coefficient, Jensen–Shannon divergence and Vertex count. The first two are different algorithms to compare the degree distribution of graphs. The last one, vertex count, is another graph attribute to measure similarity. Furthermore, users are also able to combine similarity measures. For example, we can compare two graphs based on their degree distributions but also

take under account their graph size. We can use some simple linear composition formula, like R = w1Rd + w2Rn, with Rd, Rn the degree distribution and vertex count similarity matrices respectively, and w1 = w 2 = 0.5. With various similarity matrices, users are capable of comparing similarities of graph in different aspects. To draw the graph of the number of total communities in different times. I first divided the dataset by month. Then for each sub dataset, I used Louvain algorithm to extract the communities from the network. Then I used Plotly to draw the graph. Since it is quite expensive to calculate the communities in different months, so I stored the related information in the database.

3. Implement local and global community detection method

For local community detection methods, they focus on the concepts of subgroup cohesiveness and mutuality, such as k-cliques and k-cores. Alternatively, the internal and external vertex and subgraph degrees can also be used to define community. I utilized OLCPM, Online Label propagation CPM, to implement local detection method. It first implements OCPM, Online Clique Percolation Method, which takes two inputs: networks modification events and the clique size K. It returns all maximal cliques of size not less than k. I utilized NetworkX, a Python library for studying graphs and networks, to create the graph. Then I used its k_clique_communities method to return k-clique communities in graph. The second step is label propagation, which sets out on the output communities of OCPM to discover the peripheral nodes. Each community spreads to neighboring peripheral nodes a label containing its identity and a weight representing the distance between this node and the community. Label propagation process is based on breadth-first search (BFS). I utilized shortest_path_length method of NetworkX to calculate the distance between nodes and communities. When all labels have been shared, nodes belong to the communities with which they have the shortest distance. If the distances between the node and communities are the same, then the node belong to all of the communities.

In terms of the global communities, they are often seen as a property of the whole network. One global community structure is based on the widely used concept of modularity. Another wide class of global community definitions relies on some similarity measure between network vertices. I used Louvain algorithm to implement global community detection. Louvain algorithm is a method to extract communities from large networks. It generally has two steps: At the beginning, each node is regarded as a community, and the weight of edges in the community is 0. For each node, the algorithm traverses all the neighbor nodes and measures the modularity gain by adding the node to their communities. It then places the node into the community that resulted in the greatest modularity increase. This process is repeated until the communities do not change. In the second phase of the algorithm, it first groups all of the nodes in the same community and builds a new network. Next, it calculates the weight of the edges between these newly generated communities and between all nodes within the communities. Then it goes to the step one for the next round. I utilized python-louvain package to implement this algorithm.

尚需完成的任务 Work to do:

1. Implement community search algorithm

The field of community search aims to provide efficient solutions for searching high-quality communities from large networks in real-time. Specifically, given a vertex q of a graph G, it aims to find a community, which contains q and satisfies the properties of connectivity and cohesiveness.

2. Predict graph operator output over multiple graphs based on the similarity matrix

For a given graph operator, similar graphs produce similar outputs. Based on the calculated similarity matrix, I could use this method to predict graph operator, like diameter, efficiently.

3. Write the documentation for the website

Users could look for the documentation, and know how to operate the system. Besides, the documentation also contains some useful information, such as default parameters and method description.

存在问题 Problems:

1. Nodes could belong to multiple communities, how to show them on the force-directed graph?

2. Currently, when users make a request to the system, some of the functions may take several seconds. As a result, users may repeatedly submit the same request, which is not working.

拟采取的办法 Solutions:

1. Judge whether the nodes appear in multiple communities. If it is the case, then assign a different color to these nodes.

2. Design a loading feature so users know that the system is processing the data they submitted.

论文结构 Structure of the final report:

1. Introduction
1.1 The point of the project
1.2 What has been did in the project and why
1.3 What has been achieved
1.4 The structure of the report
1.5 Hint some of the conclusion
2. Background
2.1 Apollo Framework
2.2 Community Detection
  2.2.1 Local Community Detection
  2.2.2 Global Community Detection
2.3 Community Search
3. Design and implementation
3.1 A web-based dynamic graph management systems.
3.2 A variety of visualization components.
  3.2.1 Data distribution map
  3.2.2 Force-directed Graph
  3.2.3 Heatmap and Dataset Space
  3.2.4 Total Community Graph
3.3 Community Structure
  3.3.1 OLCPM
  3.3.2 Louvain
  3.3.3 Local Search of Communities
4. Results and discussion
  4.1 The test and discussion of the visualization components
  4.2 The test and discussion of the community structure
  4.3 The test and discussion of the whole system
5. Conclusion and further work
6. References (Bibliography)
7. Acknowledgements
8. Appendices

北京邮电大学 本科毕业设计（论文）教师指导记录表

Project Supervision Log

学院 School	International School	**专业 Programme**	**e-Commerce Engineering with Law**		
姓 Family name	Yu	**名 First Name**	Weiqiang		
BUPT 学号 BUPT number	2016213346	**QM 学号 QM number**	161196754	**班级 Class**	2016215113
论文题目 Project Title	The implementation of temporal graph management system				

Date: 04-11-2019
Supervision type: face-to-face meeting
Summary: discussed the project specification

Date: 11-11-2019
Supervision type: email
Summary: received written feedback on the draft specification

Date: 20-11-2019
Supervision type: face-to-face meeting
Summary: discussed the current progress and problems

Date: 03-12-2019
Supervision type: face-to-face meeting
Summary: discussed the current progress and problems

Date: 02-01-2020
Supervision type: face-to-face meeting
Summary: discussed the early term project report

Date: 07-01-2020
Supervision type: face-to-face meeting
Summary: discussed the work plan in the winter holiday

Date: 10-02-2020
Supervision type: email
Summary: discussed the current progress and problems

Date: 17-02-2020
Supervision type: email
Summary: discussed the current progress and problems

Date: 20-02-2020
Supervision type: email
Summary: discussed the midterm progress report and slides

Date: 21-02-2020
Supervision type: email
Summary: discussed feedback about midterm progress report

Date: 02-03-2020
Supervision type: email
Summary: discussed the current progress and problems

Date: 09-03-2020
Supervision type: email
Summary: discussed the current progress and problems

Date: 16-03-2020
Supervision type: email
Summary: discussed the current progress and problems

Date: 23-03-2020
Supervision type: email
Summary: discussed the current progress and problems

Date: 30-03-2020
Supervision type: email
Summary: discussed the current progress and problems

Date: 06-04-2020
Supervision type: email
Summary: discussed the draft report

Date: 13-04-2020
Supervision type: email
Summary: discussed feedback about draft report

Date: 21-04-2020
Supervision type: online meeting
Summary: mock viva