INTERNATIONAL
HELLENIC
UNIVERSITY

# Knowledge Discovery in Heterogeneous Information Networks

**Student Name: Paraskevas Koukaras**

SID: 3301150004

SCHOOL OF SCIENCE & TECHNOLOGY

A thesis submitted for the degree of

*Master of Science (MSc) in Information and Communication Systems*

DECEMBER 2016

THESSALONIKI – GREECE

INTERNATIONAL
HELLENIC
UNIVERSITY

# Knowledge Discovery in Heterogeneous Information Networks

**Student Name: Paraskevas Koukaras**

SID: 3301150004

| | |
|---|---|
| Supervisor: | Dr. Christos Berberidis |
| Supervising Committee Members: | Associate Prof. Apostolos Papadopoulos<br>Assistant Prof. Christos Tjortjis |

SCHOOL OF SCIENCE & TECHNOLOGY

A thesis submitted for the degree of

*Master of Science (MSc) in Information and Communication Systems*

DECEMBER 2016

THESSALONIKI – GREECE

# Abstract

This dissertation was written as a part of the MSc in ICT Systems at the International Hellenic University. Graph mining techniques were utilized for modelling and discovering knowledge from various Information Networks.

Most Information Networks are usually assumed to be homogeneous where nodes are objects of the same entity and links represent relationships between them. This work deals with heterogeneous networks which seem to represent more ideally the real world networks where nodes and relations are consisted of different typed entities.

During the implementation, the first goal is the collection and selection of appropriate data and their modelling as a multi-type network. Next, the graph structure which was created in the first step will be mined for producing knowledge. Mining goals will focus on: a) Standard graph metrics (centrality, closeness, etc.) and b) utilization of hybrid algorithms for mining tasks such as ranking and clustering.

OrientDB is used as a DBMS for achieving our goals, which offers a native Java API, implementing algorithms to perform the modelling and analysis tasks for reaching our experimental analysis results.

Although this project is performed mostly solely as part of the attendance and acquisition of my MSc Degree, it would not have been possible without the continuous support and guidance of my supervisor.

I am highly indebted to Dr. Christos Berberidis, for his guidance and constant supervision as well as for being there always with all kinds of information required to complete this dissertation.
I would also like to express my gratitude towards my parents and brother for their kind support and encouragement which helped me in my endeavor to complete this work.

Paraskevas Koukaras

PKoukaras

23/12/2016

# Contents

## List of Tables

## List of Figures

# 1 Introduction

Nowadays Big Data invades more and more in our everyday lives. Most of the people that dwell in technologically advanced societies live their routine using the social media to communicate, connect, share and interact with other users.

We are becoming producers of big data at an unimaginable rate. Thus, new opportunities arise for research and development having all this data at our disposal. But, although we seem to have a seemingly un-limitless access to data, it does not mean that we can easily take advantage of it.

During the past decades, we were dealing with traditional data and data mining techniques that accompany these studies, but complex multi-typed information networks are different and new studies should take place. We need updated versions of algorithms or new computational methods if we want to effectively mine this new kind of data.

For example, social media mining is a new, fast developing and growing field which should deal with noisy, free-format and sometime times very lengthy data and all types of multimedia. Relations or linking between entities or other social networks should be utilized, using this ocean of social data, making the use of statistical and mathematical methodologies an inseparable part of the procedure of social mining techniques for harnessing knowledge.

Lately, more and more researchers and communities deal with evolving matters and studies of social and computer sciences. Rapidly evolving literature is being presented day by day showing the great interest and lusting need for disciplines and new tools for handling this kind of data aiming on producing highly acceptable insights. These insights can be very useful for other fields of sciences.

This dissertation aims on noting novel literature and methods, along with some coding, transforming data in appropriate graph structures for implementing fundamental concepts and algorithms used for complex structure mining.

To cope with these high standards, individuals in this fast-growing topic, need to have good knowledge on different fields of computer science such as machine learning, text mining, information retrieval, and social network analysis. As mentioned earlier, intensifying consulting of state of the art research papers is mandatory to effectively deal with information network mining.

*Chapter 3* contains relevant research topics that were found to be important to help address the requirements of this dissertation. Furthermore, reference to crucial research papers was made taking into consideration that the reader should understand basic information network mining concepts.

Social media users tend to generate data even if they do not understand it, noting the huge potential for business development opportunities that this research field has. Sharing information using social media makes this possible by trying to understand and show us influences in this complex world and what relations exist between different ontologies that might be hidden. Studying social media mining techniques might help enterprises serve better their customers, governments deal with important matters more effectively and so on.

# 2 Overview

This dissertation can be split into three (3) modules, the [Literature Review], [Network Schema and Data Structure, Implementation, Tools] (chapters 4-6) and the [Results-Conclusion] after the modeling-mining procedure we followed.

*Literature review* outlines necessary graph mining concepts and tasks, research, techniques on this topic providing an understanding of the fundamental elements and issues of complex information network mining (focusing on HINs). This topic is considered a challenging and a relatively new one. Thus, heavy endeavor was made trying to approach common issues. History review is not presented thoroughly, although a light consultation of relevant history notes [2] [3] [4] was made, from which a categorization of social media transpires: *internet blogs, collaborative projects, social networking sites, content communities and virtual game and social worlds*. A more understandable and yet commonly used definition considers that social media simply consists of *social atoms, entities, and interactions* between them [5]. For understanding information network and social media concepts, references to various other information technology fields were accumulated such as *machine learning* [6], *pattern recognition* [7], *network science* [8], *network analysis* [9] and *web mining* [10]. It is very important that we can handle and analyze effectively multimedia data [1] in social media networks.

*The Network Schema and Data Structure, Implementation, Tools* module shows the procedure followed for performing information network mining tasks along with analytical commendation on output. Dataset selection, explanation of network schemas, implementation of algorithms for mining tasks and tools used are discussed.

The *Results* module presents the aim of this dissertation and the fruitful process of data mining techniques, providing usable knowledge for future work; which was spotted during advising cites, as well as additional work that could expand this dissertation.

# 3 Literature review

This chapter is a concise and thorough review of the relevant literature, focusing on the most significant points of interest.

## 3.1 Social Media

The work presented in [3] and [4] provide some introductory concepts on social media, including historical elements that are useful to better understand the general context of this work.

Social media is a relatively new field of studies with lots of prospects on working new wonders. We could easily allege that this rising trend allow us to merge theoretical concepts into practical solutions that could shake the whole inception of the virtual world with the real. From social theories, there is the *Social Atom* as an individual that interacts with the *Social Molecule* which is the community, constructing seven (7) probable building blocks (*Identity, Conversations, Sharing, Presence, Relationships, Reputation, Groups*) of social media [11]. Also, a categorization of social media sites such as *blogs, social media sites, virtual games worlds* was found in [2].

Social media data is quirky, meaning that it is not like the conventional data we are used to handle in the classic data mining methodologies and can only be handled using innovating data mining techniques. We should deliver knowledge that comes from user-generated data (content) which is nearly always overwhelmed by social relations. Thus, we are in great need to develop through extensive study, techniques that would help us to achieve that task in this emerging field of data mining.

As a definition for *social media Mining* we consider: *the process of representing, analyzing and extracting actionable patterns from social media data* [15]. To do that, we use new algorithms or modified ones which come from a blend of various methodologies and principles of other science disciplines such as statistics, mathematics, computer science, sociology, machine learning etc. The data we handle is enormous in terms of size, which means that we need individuals that explicitly focus on these kinds of studies.

Social media mining scientists, need to have the necessary skills to preprocess, measure, model and find patterns in social media data, visualizing them in a meaningful manner.

This means a blend of well-established computational and social theory background along with analytical and coding skills are essential for achieving in these high-demanding tasks.

The following table represents the types of social media available. The classification is based on purpose and function. We distinguish nine (9) types of social media [3]:

Table 1: Social Media types

| Social Media type | Description | Example |
|---|---|---|
| **Online social networking** | Web-based services that allow individuals and communities to connect with real world friends and acquaintances online. Users interact with each other through status updates, comments, media sharing and messages. | Facebook, Myspace, LinkedIn |
| **Blogging** | Journal-like website for users, to contribute textual and multimedia content, arranged in a reverse chronological order. Blogs are generally maintained by an individual or by a community. | Huffington Post, Business Insider, Engadget, WordPress.com |
| **Micro-blogging** | Same as blogs but with limited content. | Twitter, Tumblr, Plurk |
| **Wikis** | Collaborative editing environment that allows multiple users to develop Web pages. | Wikipedia, Wikitravel, Wikihow |
| **Social news** | Sharing and selection of news stories and articles by communities of users. | Digg, Slashdot, Reddit |
| **Social bookmarking** | Allows users to bookmark Web content for storage, organization, and sharing. | Delicious, StumbleUpon |
| **Media sharing** | Sharing of media on the Web including video, audio, and photo. | YouTube, Flickr, UstreamTV |
| **Opinion, reviews and rating** | The primary function of such sites is to collect and publish user submitted content in the form of subjective commentary on existing products, services, entertainment, businesses and places. | Epinions, Yelp, Cnet |
| **Answers** | Platform for users seeking advice, guidance, or knowledge to ask questions. Other users from the community can answer these questions based on previous experiences, personal opinions, or relevant research. Answers are generally judged using ratings and comments. | Yahoo! answers, WikiAnswers |

Having presented the most common social media types, Table 2 follows, giving a glimpse of the user growth [17] on social media information networks.

Table 2: Examples of Social Media and user growth

| Social Media | User growth |
|---|---|
| **Facebook** | Approximately 1.71 billion active users. |
| **Twitter** | Almost 313 million active monthly users. |
| **YouTube** | More than 4 billion videos are viewed per day and 400 hours of videos are uploaded every minute. |
| **Flickr** | Number of unique U.S. visitors from January 2012 to July 2015 is 17.49 million. |
| **Wikipedia** | In January 2002, the number of Wikipedians with more than 10 Wikipedia contributions was 431 and grew to 2.14 million in January 2016. |

## 3.2 Communities and Interactions

Communities also known as clusters, groups, subgroups, are an essential concept in complex information network (e.g. social media) mining due to their potential to optimize the organization of the robust amount of entities we ought to handle. Communities can be either *emic* (explicit) or *etic* (implicit) [21]. In addition, community detection focuses on etic communities that we discover in social media sites.

There are two (2) types of community detection algorithms:

The first type is the member-based community detection, in which individuals that belong to a community become a group, depending on three distinct (3) measures *degree*, *reachability* and *similarity*.

- For *degree* measure, we find the *clique* [20] terminology which can constitute a community itself (clique might be a small community). There are also methods for identification of cliques such as *Brute-force clique identification* [19] as well as methods for handling the cliques like *k-Plex* [18] and *clique percolation* algorithm [18].
- For *reachability* measure, there are methods for subgraphs like *k-clan, k-club, k-clique* [22].
- For similarity measure, we use *Jaccard* [23] and *Cosine* [24] similarities.

The second type is the group-based community detection, which is accompanied by methods for detecting communities. The communities can be *balanced, robust, modular, dense* and *hierarchical*.

- Balanced communities are spotted by using *spectral clustering* [25].
- Robust communities are formed from subgraphs that tend to have a strong link between them. Examples of forming robust communities are *k-edge and k-vertex* [26].
- For modular communities, we utilize *modularity maximization* [27].
- For dense communities, *quasi-cliques* [28].
- For hierarchical clustering, there is the *Girvan-Newman algorithm* [29].

Indicative methods for cluster quality measuring that can be used are *Davies-Bouldin* [30], *Rand index* [31], *Silhouette index* [32].

## 3.3 Heterogeneous Information Networks

Most of the research that was conducted since lately, dealt social networks as homogeneous networks meaning that there was no differentiation of nodes and links. This does not reflect to the real networks where multi-typed components are constantly interacting with others.

The new wave of research focuses on Heterogeneous Information Networks which seem to be more promising in representing multi-typed and interconnected data without losing their semantic meaning after a data mining analysis.

As mentioned earlier, Heterogeneous Networks are characterized by their richer structure and semantic information compared with the Homogeneous. Having more fruitful data means that more opportunities arise after analysis of these networks. On the other hand, richer data come with more issues during handling them as well. The mining tasks are many and some of them will be discussed in this dissertation.

In Heterogeneous Information networks the interacting entities or components are bound to interconnected networks forming a structure of multilayer networks [34]. Our aim is to project the best way possible, real systems or networks which are highly populated by multi-typed components [35] which might be bio networks, other systems or simple social activities.

To achieve that, we need to perform the task of information network analysis, which recently has gained high attention from the research communities. We need to perform that task preserving the structural generality of the networks [36]. The concepts of link mining and analysis, social network analysis, hypertext and web mining, network science and graph mining [33] come into play enabling us to effectively perform this kind of analysis.

The nodes and links of heterogeneous networks are extremely rich in information compared with the homogeneous ones, so mining tasks such as *clustering* [40], *classification* [41] *and similarity search* [39] should be addressed reforming or updating currently available implementations. Advanced tasks deal with mining patterns from link associations, making assumptions regarding the type of objects and links, as mentioned in [36].

The concept of heterogeneous information network was first proposed in 2009 [37], which was followed by the *meta path* concept two years later in 2011 [38]. Since then, researchers around the world have made a great endeavor rapidly evolving the information retrieval techniques. Implicationally, the research world is being drawn to study heterogeneous networks which is becoming an even more interesting topic.

## 3.4 Modeling

Trying to model complex information networks might become a very difficult task. When we handle real world data we find ourselves trying to understand a chaotic mix of physical and abstract data entities which are connected forming huge inter-connected multilayer networks.

We must structure all this raw data and form interactions between interconnected multiple types forming the so called semi-structured heterogeneous information networks [42]. Real world applications handle information that deal with big data, social networks, medical, e-commerce, engineering datasets which can be modelled using heterogeneous information network techniques, enabling us to squish out useful information more effectively.

New principles regarding modeling already started to develop rapidly, studying and implementing very powerful algorithms specialized in farming interconnected data such as clustering and classification, meta-path-based similarity search and mining relations. These concepts are being studied and presented with experimental results in [42] where *RankClus, NetClus, GnetMine, RankClass, PathSim* and more are explained.

Next, the three (3) most known models (*The random graphs*, *the small world model* and *the preferential attachment*) for generating networks from real world ones will be discussed, along with real world network characteristics. A rather abstract point will be made on these, since our aim is to cover multiple concepts concerning complex information networks and social media mining.

### 3.4.1 Real world networks

Every network that exists has some characteristics. During the task of designing a network model we must take into consideration these characteristics. Our accuracy is very much dependent on how well we can mimic real world situations with artificial ones.

Since we deal with inter-connected networks, our first task is to find attributes and draw statistics from them which we find that are common amongst most of the other interconnected networks.

The measures we can use are *degree distribution*, *clustering coefficient* and *average path length* [43]. Degree distribution shows the way node degrees are scattered within a network. Clustering coefficient gives the transitivity measure of each of the networks we examine. Finally, average path length (shortest path) [44] shows the average distance that pairs of nodes have.

### 3.4.2 Random Graphs

Modelling with random graphs we assume that all links (edges) that connect nodes (individuals) generate random friendships. But that assumption does not necessarily mirror the real world situations, e.g. not always people are characterized by friendly relations.

Thus, using this technique we take a leap of faith hoping that the relations random graphs form, will generate networks that match with the real ones. We find that they exhibit a Poison degree distribution, a small clustering coefficient and a realistic average path [43].

### 3.4.3 Small World Model

The small world model comes as an upgrade, fixing some of the random graph model issues regarding the efficient real world representation. The small-world model was proposed in 1997 [46] as an improvement in suffering of clustering coefficient that comes with the random graphs. This phenomenon (small world) was furtherly discussed this period [47].

Most commonly social media individuals have limited number of connections like close friends, family members etc. This model assumes that the number of connections is equal for all members, meaning that all individuals have the same number of neighbors. Still, this seems unrealistic although it helps model more precisely the clustering coefficient that is found in real world networks.

But, even that is not precise enough, since the small-world model is found to produce a degree distribution almost the same as the Poisson degree distribution being found in random graphs [43].

### 3.4.4 Preferential Attachment Model

The preferential attachment model comes as the most promising one of the other models discussed in network modelling literature and it was proposed in 1999 [48]. Simply stated, this model assumes that new nodes that are added to networks have the tendency to connect to already existing nodes because others are already connected to them. Statistically speaking the higher the node's degree, it is more probable that new nodes will be connected to that node [43].

This kind of network forming, is called an aristocrat network which is found to be more realistic in average path lengths but still the clustering coefficient is smaller, which characteristic is unrealistic compared with the real world networks.

## 3.5 Mining

### 3.5.1 Data Quality

Before proceeding with the task of utilizing data mining algorithms, we must make sure that the data are at a certain level of quality that fits our needs. To do so, the standard requirements are categorized [5] as:

#### Noise

It highly affects [50] the performance of data mining algorithms as the process of data mining is executed having incorrect or non-representing data. It is inevitable to have real world expected results when noise is very high. But, since it is practically impossible to avoid noise we use methods [52] to mitigate the effect. Generally, we observe two (2) types of noise, the implicit errors which are introduced by tools, such as input from different types of sensors or monitoring tools; and random errors introduced during the batching processes or individuals responsible for data input, e.g. digitalization of documents [51].

#### Outliers

They represent data instances that are different (slightly or considerably) from others within the same dataset. These instances may have a more significant weight on the results when running an algorithm distorting the expected results. For example, in a dataset of people and assets, few insanely wealthy people who have significantly more assets than the average person will influence a relevant query. So, sometimes we need to remove the outliers or even keep them, always deciding depending on what is more appropriate for our data mining task. Performance of the methods [53] commonly used, varies from case to case, and outlier detection can be a difficult task [54].

#### Missing Values

Having missing values in our dataset is a very common incident. We usually fill in forms, forgetting to input all the fields, or we create accounts without registering all requested information. We should find ways to handle these missing values [55]. To mitigate this issue, we have three (3) options to choose from.

- Exclude the missing attributes from the data mining algorithm’s functionality (ignore missing values),
- Completely remove dataset instances that have missing values (instance removal),
- Populate missing value instances with the most common ones in the dataset (missing value replacement).

#### Duplicate Data

We face this phenomenon when our dataset is populated by instances having the same feature values. Thus, the problem of detecting and eliminating duplicated data is very disconcerting, and is certainly one of the biggest problems in the field of data cleaning and data quality [56]. Once again, depending on what are our mining tasks and our data warehouse, instances can be removed or kept. For instance, duplicate tweets or posts on social media are instances of duplicate data problem.

### 3.5.2 Data Preprocessing

After having verified that the quality of data is adequately matching our needs, the next procedure is the data preprocessing. Standard preprocessing tasks are described below:

1. *Aggregation*. We perform the task of aggregation when we need to combine continually varying features with multiple instances into one. An example could be, the storing

of a photo in Facebook. We could store the image's width and height or simply store the image dimension as one entry (width_height). This could aid preserving storage capacity, but also provide resistance to data noise.

2. *Discretization*. With this process, we simply decide the ranges of continuous features that we will transform to discrete ones. For example, a feature like height that characterizes a person can be split into discrete values such as tall, normal, short. In addition, it is very important to decide the actual range band of each discrete value with special attention, as it will highly affect our mining task.

3. *Feature Selection*. Depending on the mining algorithms we use and whether this algorithm takes advantage of the feature selection process, we can boost the performance of data mining task. But, not always the features in our datasets can be used. This happens since, some features are irrelevant with the task at hand. For example, when we want to predict if a person dwells in Greece, a feature "name" would be of minor importance or no important at all for our prediction task. Also, feature selection might require huge amount of computational power (affecting the time of our predictions) if we are to run tests on all features on a dataset, thus the selection should always be made carefully.

4. *Feature Extraction*. It is the procedure of the conversion of already available features to a new set of features. This conversion happens on the data and then new features are extracted that are used for the mining task. The aim of this process is to surpass the performance of an algorithm that uses the default features. If this does not happen, feature extraction is not performed correctly or it is not needed. An example could be a dataset that contains the features about vehicles and we convert the vehicle's speed and time features into a new one called distance (distance= speed * time). Also, feature extraction is like aggregation described earlier.

5. *Sampling*. A very useful and common method for saving time on knowledge extraction from a dataset. It is already mentioned that time and computational power are very important for any mining task and this is mainly due to the huge datasets we deal with. Furthermore, we discuss about social media networks where multimedia datasets take place along with large processing streams that accompany them. Sampling comes as a universal tackler for this problem, where a smaller random (in terms of ideally manageable) sample from the whole dataset is selected and the mining task is performed on that. Thus, the whole trick on effective sampling is the method that utilizes the random

sampling selection. It is of imperative need that the sample we concluded after the process, realistically (or closely) represents the whole dataset. For sampling, we distinguish three (3) techniques:

- *Random Sampling.* This is the default sampling method where instances in a dataset all have the same probability of being selected for the sample. There is a uniform selection of instances.
- *With or without replacement.* In sampling with replacement each instance from the initial dataset can be selected many times and can be added to our new sample dataset. The process without replacement is exactly the opposite of random sampling, meaning that, in case an instance is selected once it cannot be added again to the sample dataset.
- *Stratified Sampling.* This is a two-step method that first uses *bins*, which are partitioned pieces of the initial dataset and then performs random sampling on a fixed number of instances of these bins. It should be noted that this method is proposed for datasets that are not uniformly distributed.

When talking about complex network mining most of our data has a network form, meaning that they have multiple subsets of nodes and edges. The sampling techniques described above can be used for selecting the appropriate nodes and edges for our task. Also, we can use seed nodes [5] which are small sets of nodes to perform sampling on:

- their connected components,
- the directly connected nodes and edges on them,
- their neighboring nodes and edges.

All that said, assuming that the preprocessing is finalized, we can proceed to the actual mining using algorithms suitable for our goal.

### 3.5.3 Data mining algorithms

The algorithms in data mining are numerous and a categorization can be made, from very specific to a highly generic one. The categorization we will present is a rather simplistic one: Two (2) categories widely known, supervised learning and unsupervised learning [5].

In the first category (supervised learning) we have the class attribute and our aim is to make a prediction on what the class attribute value could be.

In unsupervised learning, our dataset has no class attribute and our aim is to find entities/instances, that have a high level of similarity and try to group them. By effectively grouping them we create other opportunities on behalf of analysis tasks, such as pattern recognition.

## Supervised learning

Supervised learning can be distinguished to *classification* and *regression*. Classification refers to the discrete class labeling that occurs to feature values. Regression is the form of classification where these labels happen to be real numbers. Common classification algorithms are *decision tree learning, naïve Bayes, nearest neighbor* [58]. Yet, these are not suitable in their native version to fulfil our needs in HINS or social media mining every time.

Supervised learning algorithms split the labeled dataset into two (2) sets, the training and the testing set to perform a consistent evaluation. There are various algorithms that perform that task, for example *leave one out* and *k-fold cross validation* used on twitter data [57].

As mentioned earlier, common classification algorithms are used for objects that have the same structure, something that is not usable for real world datasets where links between entities exist.

To satisfy these needs, linked based object classification comes to play, supported by graph theory forming nodes linked with others via edges. A new wave of research has been conducted extending the capabilities of common classification methods [59].

In homogeneous networks our nodes and links are identical, on the other hand, in heterogeneous information networks other demands exist.

- The nodes do not have the same structural type allowing us to apply methods that classify many different types at the same time.
- Labels expand through links and spread into various dissimilar nodes.

For this type of networks, heterogeneous information networks can be considered when viewed macroscopically as a vast ocean of knowledge that is being transferred by waves of information. This specific incident, triggers the opportunity to visualize and predict knowledge flow between dissimilar objects and links.

Many researchers experiment on rectifying the common classification methods to meet the needs of heterogeneous information networks. A list of some of these novel algorithms along with their functionality description follows:

- Transductive classification, is the prediction of labels of unlabeled data.
- Inductive classification, is the generation of a decision function that decides based on data spread to the whole data space.
- Multi-label classification, deals with simultaneous multiple labeling.
- Rank-base classification, is a hybrid method for utilizing both ranking and classification at the same time.
- Information propagation with classification, another hybrid utilizing propagation with classification procedures.

Table 3 contains examples of the categorization of mining tasks [33].

Table 3: Supervised learning novel methods

| Mining Task | Method/Research Paper | Description |
|---|---|---|
| **Transductive classification** | GNetMine | Models the link structure in information networks with arbitrary network schema and arbitrary number of object/link types. |
|  | HetPathMine | Performs clustering with small labeled data on HIN through a novel meta path selection model. |
| **Inductive classification** | IMBHN | Establishes a classification model assigning weights to textual terms (in a bipartite HIN) representing textual document collections. |
| **Multi-label classification** | Graffiti | Performs multi-label graph-based classification for modeling the mutual influence between nodes as a random walk process. |
|  | Kong et al.[61] | Utilizes multiple types of relationships drawn from the linkage structure for multi-label classification. |
|  | Zhou et al. [62] | Performs edge-centric multi-label classification approach considering both the structure affinity and the label vicinity. |
| **Ranking-based classification** | RankClass | Utilizes a framework to perform more accurate analysis. |
|  | F-RankClass | Utilizes classification framework that can be applied to binary or multi-class classification of unimodal or multimodal data. |
| **Information propagation-classification** | Jendoubi et al. [63] | Classifies social messages based on self - spreading in the network and the theory of belief functions. |

## Unsupervised learning

Unsupervised learning deals with clustering which is the process we follow to create groups from our datasets. These groups are consisted of objects or entities that are characterized by the same or identical attributes, but are adequative different from entities that belong to other clusters. For running a clustering algorithm, we need to specify the distance measure (e.g. Euclidean, Manhattan, Jaccard, Coisine distances) [65].

Other key points of common clustering methods are the process of object selection and the methods used for the evaluation of clustering results [66]. So, for evaluating results we can use quality measures like *cohesiveness* (measure for object-to-object distance), *separateness* (measure for cluster-to-cluster distance) and *silhouette index* (mix of cohesiveness and separateness) [67].

But for studying clustering on networked data, things change considering that data is most commonly stored in homogeneous information networks, where community handling comes to play. Most of these methods are based on multiple subgraph generation and handling (e.g. normalized cuts [68]); much research is being conducted optimizing this procedure [69] and solving arising problems [70].

What is more, heterogeneous information networks with the ability to handle entities of different types as well as multi-typed links expand even more our ability to handle more packed data.

Despite the vast amount of information that can be stored in heterogeneous information networks and difficulties that arise, they allow to store much more information and develop considerably more methods for learning.

To give a glimpse of the tasks being developed lately (based on clustering), Table 4 depicts some of the novel models/algorithms for unsupervised tasks in heterogeneous information networks [33].

Table 4: Unsupervised learning novel methods

| **Task** | **Method/Research Paper** | **Clustering algorithm description/characteristic** |
|---|---|---|
| **Creation of balanced communities** | Aggarwal et al. | Introduction of local succinctness property. |
| **Calculation of incomplete attribute information and network structure information** | Sun et al. | Model-based clustering algorithm. |
| **Modeling structure and content of social media networks using outlier links** | Qi et al. | Implementation based on heterogeneous random fields. |
| **Tackling of community detection problem** | Cruz et al. | Integration of structural and compositional dimensions for generating attributed graphs. |

| | | |
|---|---|---|
| **Clusters detection based on connections and vertex attributes of the network** | TCSC | Density-based clustering model. |
| **Topic modeling of HIN in a unified way** | Deng et al. | Topic model with biased propagation. |
| **Cluster identifier** | LSA-PTM | Link-topic propagation between multi-typed entities for clustering. |
| **Unified topic model** | Wang et al. | Topic mining and multiple entities clustering. |
| **User guided cluster generation using path selection** | Sun et al. | Semi-supervised cluster algorithm. |
| **Cluster generator** | RankClus | Clustering of specific objects in a bipartite network based on qualities of clustering and ranking are mutually enhanced. |
| **Cluster generator** | NetClus | Handles clustering in a star-schema network. |
| **Clustering and ranking** | ComClus | Applies star schema network to merge the heterogeneous and homogeneous information. |
| **Rank-based clustering** | HeProjI | Arbitrary schema on HIN by projecting the it into a sequence of subnetworks. |
| **Clustering and ranking** | Chen et al. | Probabilistic generative model on heterogeneous network with arbitrary schema. |
| **Clustering and ranking** | Wang et al. | Automated Construction of multi-typed topical hierarchies. |
| **Community detection and ranking in directed HIN** | OcdRank | Community detection and community-member ranking. |
| **Outlier detection** | Gupta et al. | Outlier-aware approach with joint non-negative matrix factorization for discovering community distribution patterns. |
| **Outlier detection** | Zhuang et al. | Uses queries and semantics for subnetwork outlier detection. |
| **Outlier measure** | Kuck et al. | Meta-path algorithm for mining outliers in HIN. |

### 3.5.4 Task Categorization

“Heterogeneous Information Networks” is considered one of the most promising topics in the new era of effective complex information networks and social media mining. The mining tasks associated with HINs should evolve to cope with the new aggressively expanding demands on this field of studies.

During the past years, a great effort has been made by writing papers, presenting methodologies, novel algorithms and implementations for these tasks. For the needs of this dissertation we found that the most appropriate categorization should be the one proposed in a very recent survey on heterogeneous networks aka HINs [33]. Next, an abstract presentation of the main mining tasks categorization follows.

### Similarity

One of the most researched and studied measures in mining. It is a way for finding out how similar objects are. It acts as the base for numerous other data mining techniques such as clustering, web searching etc. There is also a categorization of this measure found in literature; feature based approach and link based approach.

### Clustering

Clustering is also a very well-known process for data mining that spreads huge dataset objects into sets or smaller groups (clusters) that have a high degree of similarity, although preserving dissimilarity with objects in neighboring clusters.

Networked data are different from conventional data studied the past decades, where the clustering was being done relying on the features of the objects within the datasets [66]. Nowadays, there is a tendency from all kind of researchers and scientists to deal networks as heterogeneous ones which makes clustering a very useful task for drawing knowledge.

### Classification

Classification comes to play when we need to calculate probable class labels, which is achievable through a generation of a classifier or a suitable new model. In machine learning, the classification is performed on identically structured objects but for our needs we must take into consideration that we must also deal with links (associations).

Thus, a linked based object classification must take place where entities (nodes) relate to one another forming data graphs. Traditional methods are often reused or extended to be able to handle this kind of connections [59].

### Link Prediction

Effective link prediction is a very concerning issue when discussing about link mining. It deals with algorithms we use that try to find out if a link between nodes exists, using as rules the following statements: a) the observation of other nodes and b) the attributes of these nodes.

A great deal of papers discusses about prediction that only takes into consideration the structural properties of social networks using predictors [73]. Other methods use the attribute information itself for link prediction [74].

## Ranking

Ranking functions are very important since they can measure an object’s importance within a social network. For example, RankClus deals with bipartite networks creating groups (clusters) of objects preserving the equality of importance both on ranking and clustering concepts [37]. NetClus, is ideal for performing clustering when the network has the characteristics of a star type schema [76]. Many others exist like HeProjI, OcdRank etc [77].

## Recommendation

Recommendation systems are consisted of a wide variety of algorithms originated from various fields of different sciences such as information retrieval, machine learning, statistics, mathematics etc. The main aim is to make recommendations to users about every kind of objects or services that might be suitable for them. To do so, similarity measures must be utilized.

The past years, recommendation systems were naïve, just measuring user specific feedback information in order to recommend. Lately, more and more smarter techniques or updated ones come into play such as collaborative filtering [78], matrix factorization [79] or circle based techniques [80] to boost to new levels the user recommendation experience.

## Information Fusion

Information fusion is one of the top priorities and tasks that highly characterize the heterogeneous information networks. It implements powerful algorithms that fuse objects from the same or different networks whether they have completely relevant or irrelevant semantic meaning. Social media networks are full of with this type of information, creating the desire for research community to deal with this topic.

The task is very difficult, having to gather all this information and merge it to effectively mine it. Information fusion takes place also in bioinformatics [81] and in web semantics research fields [82]. Our aim is to fuse information of many different heterogeneous information networks and improve the elaboration and contemplation of the knowledge we receive.

## Measures

This section discusses about measures we can use for HINs and social media networks. Each measure gives useful feedback relevant with its specialization. An abstract presentation of the most fundamental measures follows.

***Centrality***

Generally, centrality is a measure that tries to calculate the central node in a graph. There are various versions of centrality developed to evaluate the importance of vertices within graphs.

*Degree centrality*, brings into play the degree value, calculating that the most central node is the one with the highest degree value [86].

*Eigenvector centrality*, is a generalized implementation of the *degree centrality* that calculates the most important node (central) as the one that has the most connections with other important nodes. For this centrality measure the value is computed using an adjacency matrix for finding its eigenvector [87].

Every measure has its pros and cons so; *Katz centrality* comes as an improvement for eigenvector centrality when dealing with directed graphs with the introduction of a bias term [89].

Google uses the *RageRank* centrality to rank webpages, and this implementation is considered a more normalized one compared with the *Katz* centrality [83].

*Betweenness* centrality, is utilized following the concept that the whole graph is formed of multiple hubs of nodes where the root of these hubs are always the central nodes [85].

*Closeness* centrality, is a measure that considers the central nodes to be marked as central because they are close to the rest of the nodes [84].

All these measures of centrality can be implemented in a more generic form (for a vaster measure implementation) where nodes are grouped together and we can distinguish group degree centrality, group betweenness centrality and group closeness centrality [88].

***Transitivity and Reciprocity***

One of the most common phenomenon in social media instance handling is the friendship between nodes aka. node linking. In our endeavor to understand the way nodes establish links in-between them, we came up with the terms of transitivity and reciprocity. Transitivity is “when a friend of my friend is my friend” and considered for closed triads of edges, while reciprocity is a simpler version, “if you become my friend, I will be yours”, considering only closed loops of length two (2) which can happen only in directed graphs.

To analyze this behavior, we use the clustering coefficient formulas [90], which can be distinguished to global clustering coefficient and local clustering coefficient. These formulas give a measure for analyzing transitivity of the whole network (global), as well as transitivity for individual nodes (local).

### *Balance and Status*

To decide whether relationships are formed in a consistent way in social media, we use some theories to perform this validation. For this dissertation, a choice amongst various theories is made and we will present just the two (2) most common ones. The first is called "social balance" and the second is "social status" [91]. An abstract description of each of these theories follows:

*Social balance,* aka "structural balance theory" most often represents consistency in social media in terms of forming rules amongst entities. An informal paradigm of such rules could be:

The friend of my friend is my friend,

The friend of my enemy is my enemy,

The enemy of my enemy is my friend,

The enemy of my friend is my enemy.

*Social status*, establishes another type of rule calculating how consistently entities assign status to the entities near them. Again, an informal paradigm of such rule is:

If X has a higher status than Y and Y has a higher status than Z, then X should have a higher status than Z.

Both theories are very important for measuring consistency between relationships of entities in social media mining. When used together can be proved a very precise tool for forming, updating or recreating relationships to improve our social mining task.

### *Similarity*

Since we deal with complex information networks and social media mining, similarity can be calculated by referring to structural equivalence. Loosely speaking, structural equivalence [12] is referring to the level of which two (2) nodes are considered similar when they have common neighboring nodes. This means that they share the same social environments as well as they might have similar attitudes and behaviors (in the form of edges or vertex attributes). For measuring similarity, we use the concepts of *Cosine similarity* [24] and *Jaccard similarity* [23].

# 4 Network Schema and data structure

This chapter offers a transition to a more practical point of view regarding information network mining on HINs. It is important to mention what type of data structures and network structures we have to model in order to extract knowledge.

## 4.1 Data structure

Generally, a heterogeneous information network can be consisted of three (3) types of data [13]:

*1. Structured data*

Structured data focus on solving the issue of efficient handling of any kind of data. They aim on taking a form that optimizes specific operation implementation as well as reducing the complexity of performing tasks. This is the reason why many types of structured data exist to match our needs.

To effectively handle structured data, we use data models that uniquely define how our data will be manipulated (stored, accessed etc.). Examples are numeric, alphabetic, date time, address and more, which normally are accompanied by restrictions regarding their length.

Their advantages usually stem from their ease of querying and analyzing them. On the other hand, nowadays data is vast and most often cannot be squeezed to follow the organized structure. Database tables are most commonly used being organized with entity- relation models.

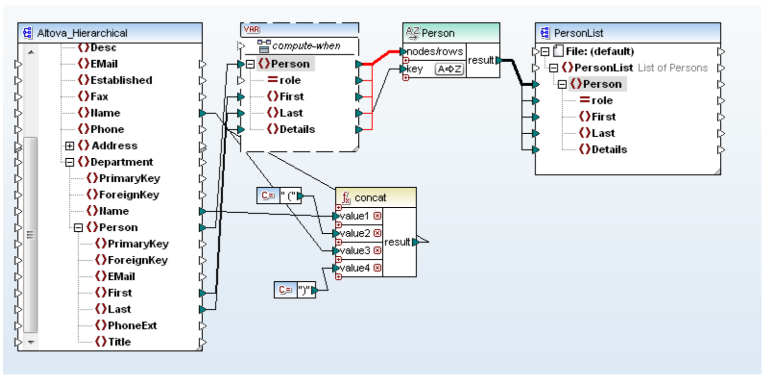


Figure 1: Example of structured data [14]

*2. Semi structured data*

Semi structured data is the type of data that keeps half of the characteristics of both other two types. Although it can be considered structured data it does not necessarily confront with a strict structure model.

Most often these data contain tags or other markers easing the process of identifying specific attributes on them. Also, they might include metadata characterizing the data that are bound to them. For example, word documents, may be labeled (metadata) showing the editor or date time of the last modification. XML as well as other markup languages are the ones most commonly used for handling semi-structured data.

Characteristics:

a. XML format data

b. object→ attribute→object

c. relation→connections among attributes

*3. Unstructured data*

This type of data is the one that cannot be classified or apply to them a formal form or structure. They are mostly text which might be consisted of dates numbers and more. They are characterized by ambiguity and irregular ways of data structuring.

Unstructured data examples are the ones that might have recognizable entities and extractable relations like photos, videos, pdf files, webpages, word documents etc.

## 4.2 Network Schema

The most commonly used HIN network schemas that might ease the mining tasks described earlier are being presented:

### 4.2.1 Multi-relational network with single typed object

The basic characteristic of this type of schema, is that the object type is only one but the relations that it can enfold are always more than one. Examples are Facebook and Twitter.

In social network theory, the existence of multi-relational schemas is very common especially for social media sites (Facebook), where we observe millions of users connecting with one another forming millions of links that might represent actions such as messaging, (video) calling, sharing, browsing and so on.

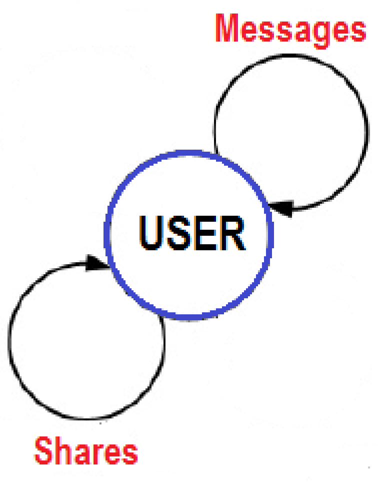


Figure 2: Multi-relation schema

Characteristics:

- Object type = 1
- Relation type > 1

### 4.2.2 Bipartite network

Another common schema observed in HIN is the bipartite network that models a relation or interaction in-between two (2) different types of objects such as multimedia-video as depicted in Figure 3.

No special explanation presented here, although there are extensions of bipartite networks modeling the existence of k-relations between objects forming links with other neighboring objects.

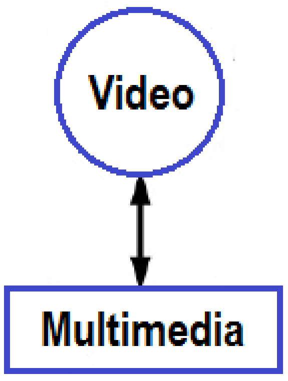


Figure 3: Bipartite schema

Characteristics:

a. Object type = 2

b. Relation type > 1

c. k-partite graph can be constructed

Examples: User-item, Document-word [33]

### 4.2.3 Star-schema network

This network schema is the most widely utilized in HIN. It represents the most common transformation of relational databases where an object that (might have attributes) generates a HIN acting as a hub where other objects can connect to.

Common examples of such networks might be any typical relational database model like the dblp database [45] which represents a bibliographic network with objects such as author, book, article etc. with links in-between them.

Also, a movies database or a business database as we will experiment on them in the implementation chapter. Figure 4 depicts a possible star-schema network showing that the Movie node acts as the hub node.

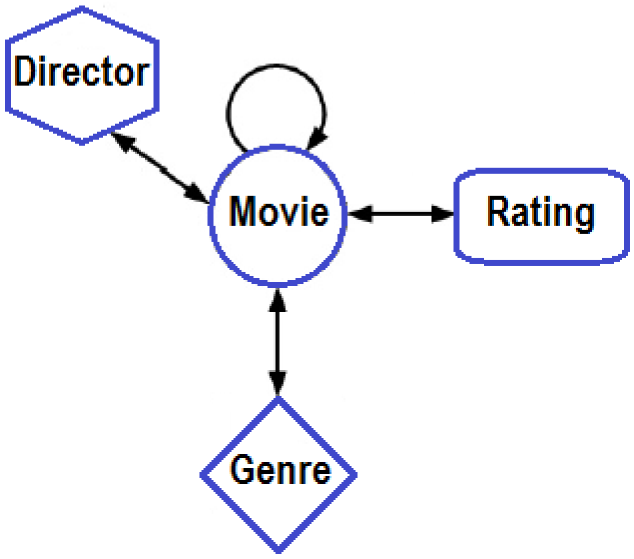


Figure 4: Star schema

Characteristics:

a. HIN that is using the target object as a hub node

Examples: Bibliographic information network, Movie network, dblp network

### 4.2.4 Multiple-hub network

Multiple-hub networks come as an improvement of star-schemas with regards to information complexity. They are used to represent even more complex network structuring, as they are consisted of many hubs. They are utilized when high specificity is required in our data representation.

Examples of implementation of such networks are common in the fields of bioinformatics. They could also be used in astrophysics or complex mathematic structures where huge fragmentation of network objects is required.

Figure 5 shows a non-realistic example of such a network structure. Although a multi-hub network for user-country-job etc. objects might stand realistically, depending on the task at hand.

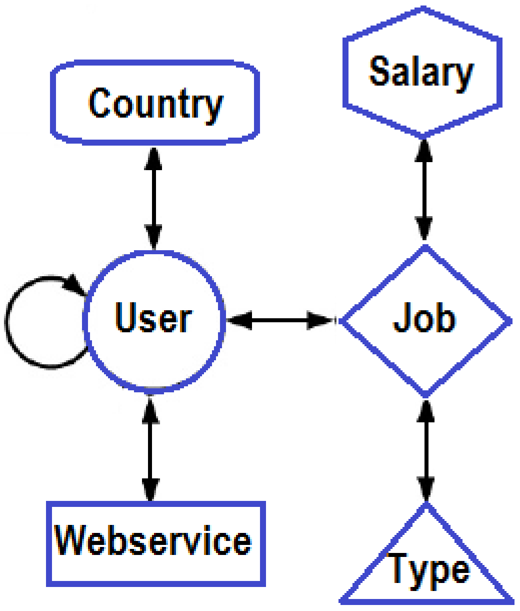


Figure 5: Multiple-hub schema

Characteristics:

a. HIN that is using multiple hub nodes

b. High (probable) structural complexity

Examples: bioinformatics data, or any other network with high specialized information network

## 4.3 Complex Information Network Data and Social Media Data

Although it might seem more appropriate to use data from social media sites such as Facebook or Twitter, for the purposes of this dissertation we will be using dataset generated for academia, research or development use.

Such datasets are the Yelp and Movielens datasets available from https://www.yelp.com/dataset_challenge/ and http://grouplens.org/datasets/movielens/ respectively. Enforcing the appropriate network schema on social media data, the mining tasks should be the same in terms of data handling.

# 5 Implementation

This chapter discusses the implementation of network schemas that will allow to utilize theories advised, along with analytical experiments conducted on different datasets [64] [60] [71].

## 5.1 Problem Definition

Most of data informational objects, individuals, groups, or components are interconnected or interact with each other one way or another without loss of generality. Such interconnected networks are called information networks.

Some of the information networks include social networks, the world-wide-web, publication networks, biological networks, and so on. Nowadays, the analysis of social networks has gained extremely wide attentions. Social media and online social network gave rise to detailed traces of human social activity which offers many opportunities to analyze and model the behaviors of millions of people.

Many models have been proposed and shown to be useful. They have been successfully applied or extended to many problems, including document clustering and classification, however, most of these models only consider the textual information while ignoring network structures.

The aim of this project is to develop a methodology for effectively implementing any given mining task, visualizing the results in a graph database where most commonly used graph measures can be calculated. This should be expanded to allow mining tasks in Heterogeneous Information Networks (HINs).

The methodology could be developed in Java or any other programming language that can be combined with a multi-model database that utilizes the characteristics of a NoSQL database management system.

To demonstrate the endeavor of the stated aim, literature associated with the problem at hand is discussed, including algorithms currently in use for mining tasks, as well as metrics/measures for deciding the most appropriate network structures or evaluating performance.

Experiments are necessary for evaluating the procedure followed, to conclude to the most appropriate method, depending on the mining task and the datasets at hand. Also, experimental results are used for gaining the first insight of issues that might arise through the process of datamining.

Focus is made on bipartite and star-schema networks generating test subjects using datasets that are being modeled to follow these network structures respectively.

*Experiment 1* utilizes the Yelp dataset forming a bipartite network consisting of two (2) types of objects, Users and Businesses. This experiment tries to utilize and create the appropriate infrastructure for implementing a NOSQL DBMS that will be able to handle any bipartite schema. The means for achieving that goal is the use of Node.js and OrientDB. It aims on utilizing modeling of large information networks such as HINs, enabling the knowledge discovery from the produced model. The knowledge discovery is partially presented and tested in the form of effective queries on the generated database.

*Experiment 2* utilizes the Movies dataset creating a star-schema network consisting of three (3) types of objects, Users, Movies, Genres. This experiment tries to utilize and create the appropriate infrastructure for implementing a NOSQL DBMS that will be able to handle any star-schema network. The means for achieving that goal is the use of NetBeans and the Graph API of OrientDB. Knowledge discovery takes the form of effectively querying the generated database and producing results. In addition, graph measures are calculated using Gephi along with its OrientDB plugins to query the Movie database in the form of http requests.

*Experiment 3* utilizes a synthetic graph of Professors-Msc Programs creating a bipartite network consisting of two (2) types of objects, Professors and MSc programs. The experiment is conducted solely in Java code in NetBeans, using the appropriate imported libraries and generated functions. The Java project implements a proposed algorithm that utilizes both ranking and clustering of objects that belong in an information network. As discussed in literature section, the most common concepts in information network analysis are clustering, classification, ranking, pattern discovery; where we want to explore the power of links (edges) and conclude to logical facts from redundant information. This experiment deals with ranking and clustering.

## 5.2 Experiment 1

Experiment 1 implements a graph representation of the Yelp dataset in OrientDB using Node.js® and the npm package (which is the largest ecosystem of open source libraries in the world) to read line by line the Yelp dataset files and create nodes and edges abiding with our proposed network schema.

Thus, the main concept of work here is reading line by line .json files and then start performing CRUD operations to populate the OrientDB database. OrientDB provides interface for querying the database using SQL, as well as numerous commands for handling our data [60].

### 5.2.1 Yelp Dataset

Description of the Yelp dataset files is supplied at the Appendices section. We used three (3) files as supplied by the Yelp challenge, yelp_academic_dataset_business.json for businesses, yelp_academic_dataset_review.json for reviews and yelp_academic_dataset_user.json for users.

The datasets report 85.902 business records, 572.448 user records and 129.763 produced edges (reviewed by) linking businesses with users.

### 5.2.2 Yelp Network Schema

The network schema for Experiment 1 is the most simplistic one we used that was aiming on evaluating the scalability and overall performance of OrientDB when handling massive bulk of data.

The graph was produced from two (2) Vertice classes, Business and User, and an Edge class reviewed_by. As it is shown in the figure below, the edges produced are outbound from Business class creating a relation Business→reviewed_by→User.

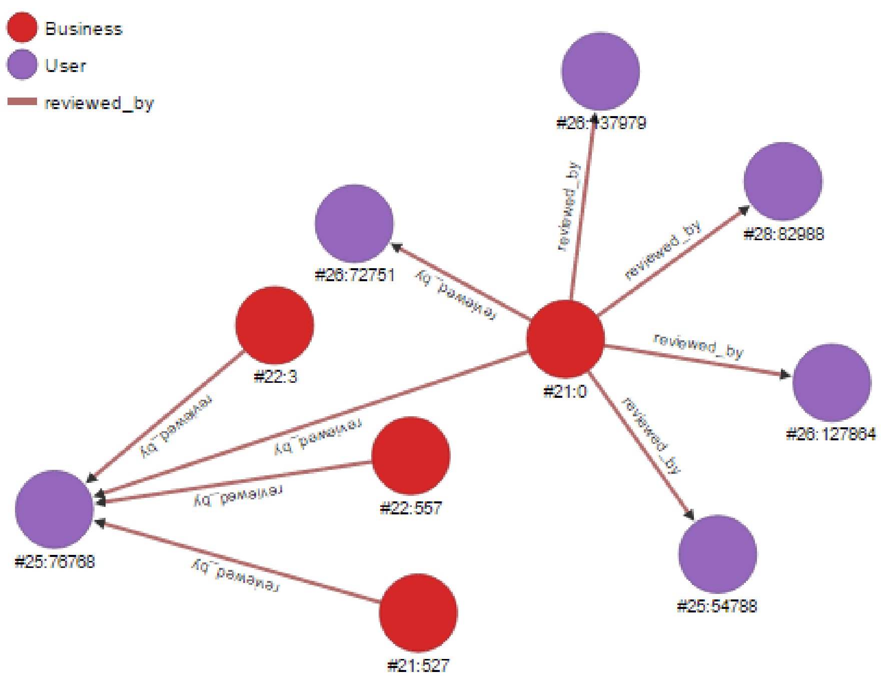


Figure 6: Network schema of Yelp database

### 5.2.3 Implementation

The implementation was achieved using the OrientDB and the Node.js. Appropriate packages where installed on a local folder using npm to achieve our main goal (database population with data), running three (3) custom .js script files that utilize the whole procedure.

These scripts are nodes.js, edges.js and orientdb-backend.js. Detailed functionality of each of these scripts is supplied in the form of comments within the code accompanying this dissertation.

A general roadmap of the functionality of these scripts is the following:

- The orientdb-backend.js is consisted of methods that allow the connection to the OrientDB server and handles the implementation of functions that create the necessary nodes, edges and perform index association for generating the appropriate links.
- node.js is a script that works as a parser reading data from a .json file and populating the dataset with the nodes requested.
- edges.js is a script that works as a parser reading data from a .json file and generating the edges requested.

All that mentioned, the network schema abides with the bipartite network schema described in Chapter 4.

According to the literature advised, bipartite is a commonly used schema in social media when the different object types in our dataset are strictly two (2). For our experiment, these objects are Businesses and Users.

Furthermore, the whole implementation structure in this experiment could be easily expanded or modified to allow the utilization of other real world datasets that follow this specific network structure.

## Querying Yelp database

After running the aforementioned scripts and successfully having populated the database in OrientDB studio, it is considered necessary to query the database to evaluate the produced product’s performance, correctness and result precision.

Some queries along with their results follow that one could use to get useful information from a database utilizing a bipartite network schema.

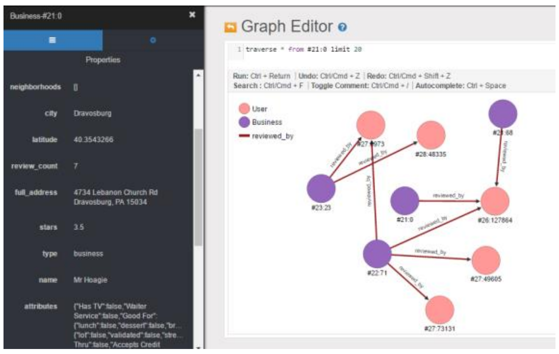
***Query #1***

*SQL Syntax:* TRAVERSE * FROM #21:0 LIMIT 20

*Description:* The first query is a rather simple traverse command, which retrieves nodes and edges (relationships) across connected records that populate the database. #21:0 is the node record from where OrientDB will start traversing. More specifically it is a business record.

*Result:* The result of the query in graph editor is shown in the figure bellow. We observe that each entry in the OrientDB has a unique index, for our example #21:0 corresponds to the Business named “Mr. Hoagie”.

Another thing to mention, we are limiting the results produced to twenty (20) using <limit 20>.



Business-#21:0

Properties

| | |
|---|---|
| neighborhoods | [] |
| city | Dravosburg |
| latitude | 40.3543266 |
| review_count | 7 |
| full_address | 4734 Lebanon Church Rd Dravosburg, PA 15034 |
| stars | 3.5 |
| type | business |
| name | Mr Hoagie |
| attributes | {"Has TV":false,"Waiter Service":false,"Good For": {"lunch":false,"dessert":false,"br... {"lot":false,"validated":false,"stre... Thru":false,"Accepts Credit |

Graph Editor

```
traverse * from #21:0 limit 20
```

Run: Ctrl + Return | Undo: Ctrl/Cmd + Z | Redo: Ctrl/Cmd + Shift + Z
Search : Ctrl/Cmd + F | Toggle Comment: Ctrl/Cmd + / | Autocomplete: Ctrl + Space

Figure 7: Query #1 Yelp database

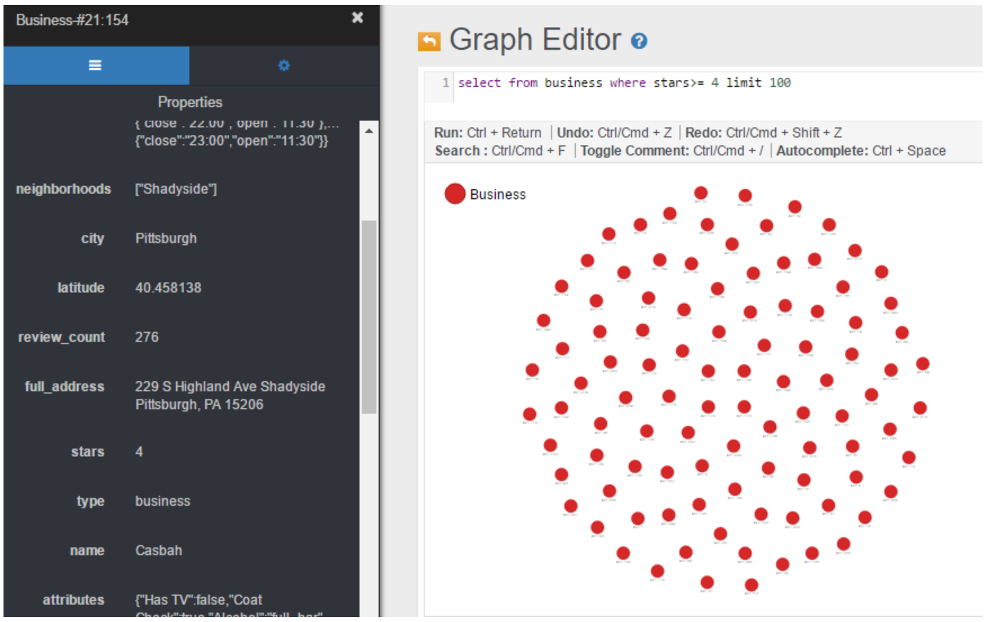
***Query #2***

*SQL Syntax:* SELECT FROM business WHERE stars >= 4 LIMIT 100

*Description:* This query is a simple select command which retrieves nodes of type business from the database. It utilizes the > = operator to be more specific to the node selection.

*Result:* The result of the query in graph editor is shown in the figure bellow. The following query returns up to 100 businesses rated with 4 or more stars.

We are limiting the results produced to one-hundred (100) using <limit 100>.



Business-#21:154

Properties

{"close":"23:00","open":"11:30"}}

| | |
|---|---|
| neighborhoods | ["Shadyside"] |
| city | Pittsburgh |
| latitude | 40.458138 |
| review_count | 276 |
| full_address | 229 S Highland Ave Shadyside Pittsburgh, PA 15206 |
| stars | 4 |
| type | business |
| name | Casbah |
| attributes | {"Has TV":false,"Coat |

Graph Editor

```
select from business where stars>= 4 limit 100
```

**Run:** Ctrl + Return | **Undo:** Ctrl/Cmd + Z | **Redo:** Ctrl/Cmd + Shift + Z
**Search :** Ctrl/Cmd + F | **Toggle Comment:** Ctrl/Cmd + / | **Autocomplete:** Ctrl + Space

Figure 8: Query #2 Yelp database

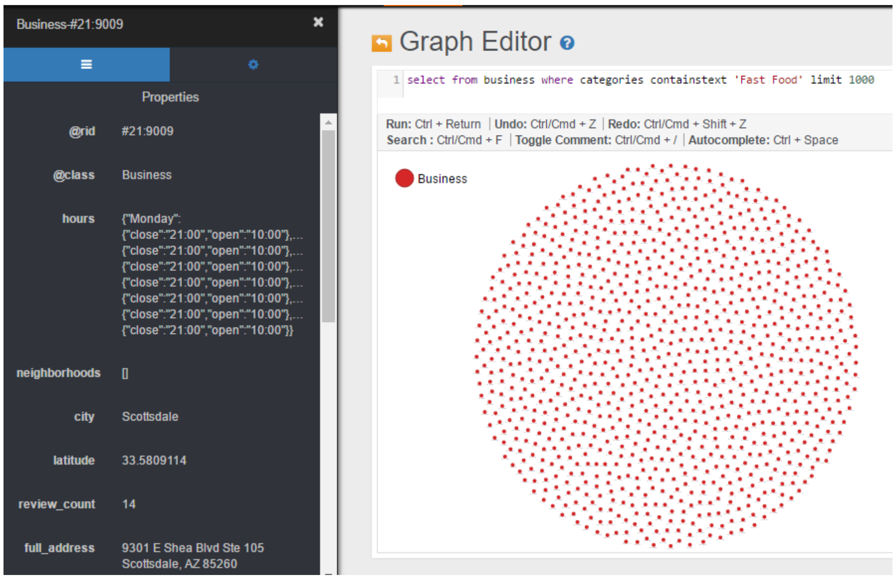
***Query #3***

*SQL Syntax:* SELECT FROM business WHERE categories CONTAINSTEXT 'Fast Food' LIMIT 1000

*Description:* This query is a simple select command which retrieves nodes of type business from the database. It utilizes the containstext function to be more specific to the node selection.

*Result:* The result of the query in graph editor is shown in the figure bellow. The following query returns up to 1000 businesses that sell fast food.

We are limiting the results produced to one-thousand (1000) using <limit 1000>.


Business-#21:9009

Properties

| | |
|---|---|
| @rid | #21:9009 |
| @class | Business |
| hours | {"Monday": {"close":"21:00","open":"10:00"},... {"close":"21:00","open":"10:00"},... {"close":"21:00","open":"10:00"},... {"close":"21:00","open":"10:00"},... {"close":"21:00","open":"10:00"},... {"close":"21:00","open":"10:00"},... {"close":"21:00","open":"10:00"}} |
| neighborhoods | [] |
| city | Scottsdale |
| latitude | 33.5809114 |
| review_count | 14 |
| full_address | 9301 E Shea Blvd Ste 105 Scottsdale, AZ 85260 |

Graph Editor

```
select from business where categories containstext 'Fast Food' limit 1000
```

**Run:** Ctrl + Return | **Undo:** Ctrl/Cmd + Z | **Redo:** Ctrl/Cmd + Shift + Z
**Search :** Ctrl/Cmd + F | **Toggle Comment:** Ctrl/Cmd + / | **Autocomplete:** Ctrl + Space

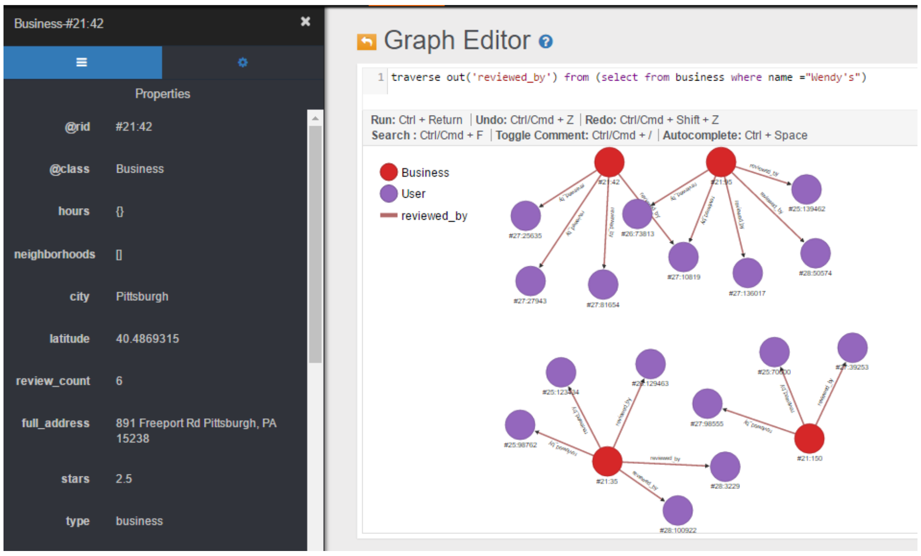
Figure 9: Query #3 Yelp database

***Query #4***

*SQL Syntax:* TRAVERSE out('reviewed_by') FROM (SELECT FROM business WHERE name ="Wendy's")

*Description:* This query is a traverse command which retrieves nodes and edges that traverse out from businesses. It utilizes a nested select as well as an out (<edge class name>) function to be more specific to the data selection.

*Result:* The result of the query in graph editor is shown in the figure bellow. The traverse Lists all users who have reviewed the business named "Wendy's". Specifically, the result also includes the businesses as well and not only the list of users who reviewed the Wendy's restaurant.


Business-#21:42

Properties

| | |
|---|---|
| @rid | #21:42 |
| @class | Business |
| hours | {} |
| neighborhoods | [] |
| city | Pittsburgh |
| latitude | 40.4869315 |
| review_count | 6 |
| full_address | 891 Freeport Rd Pittsburgh, PA 15238 |
| stars | 2.5 |
| type | business |

Graph Editor

```
traverse out('reviewed_by') from (select from business where name ="Wendy's")
```

**Run:** Ctrl + Return | **Undo:** Ctrl/Cmd + Z | **Redo:** Ctrl/Cmd + Shift + Z
**Search :** Ctrl/Cmd + F | **Toggle Comment:** Ctrl/Cmd + / | **Autocomplete:** Ctrl + Space

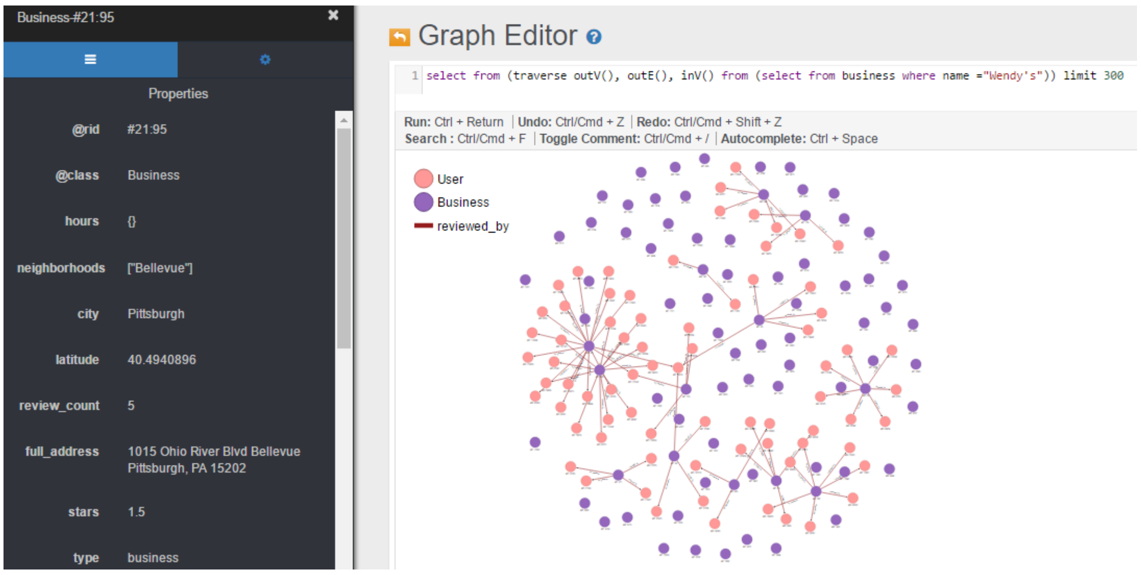
Figure 10: Query #4 Yelp database

***Query #5***

*SQL Syntax:* SELECT FROM (TRAVERSE outV(), outE(), inV() FROM (SELECT FROM business WHERE name ="Wendy's")) LIMIT 300

*Description:* This query is a select command which lists nodes and edges that traverse out from businesses. It utilizes a nested traverse as well as a plethora of Vertice and Edge class functions and another nested select to be more specific to the data selection.

*Result:* The result of the query in graph editor is shown in the figure bellow. It traverses each edge review and node user who reviewed the business named “Wendy’s”, limiting the results to three-hundred (300).



Business-#21:95

Properties

| | |
|---|---|
| @rid | #21:95 |
| @class | Business |
| hours | {} |
| neighborhoods | ["Bellevue"] |
| city | Pittsburgh |
| latitude | 40.4940896 |
| review_count | 5 |
| full_address | 1015 Ohio River Blvd Bellevue Pittsburgh, PA 15202 |
| stars | 1.5 |
| type | business |

Graph Editor

```
select from (traverse outV(), outE(), inV() from (select from business where name ="Wendy's")) limit 300
```

**Run:** Ctrl + Return | **Undo:** Ctrl/Cmd + Z | **Redo:** Ctrl/Cmd + Shift + Z
**Search :** Ctrl/Cmd + F | **Toggle Comment:** Ctrl/Cmd + / | **Autocomplete:** Ctrl + Space

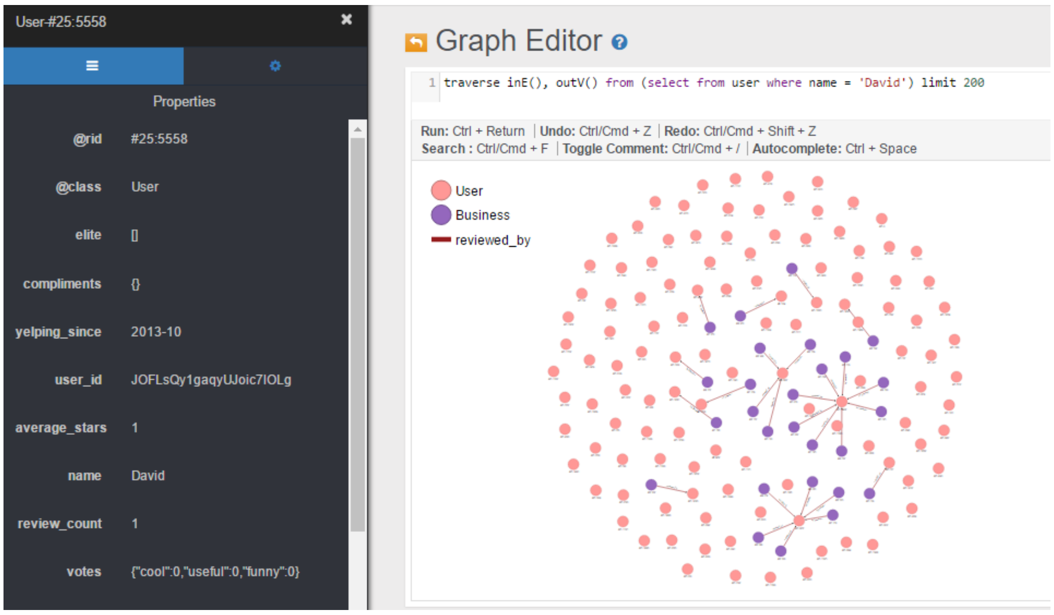
Figure 11: Query #5 Yelp database

***Query #6***

*SQL Syntax:* TRAVERSE inE(), outV() FROM (SELECT FROM user WHERE name = 'David') LIMIT 200

*Description:* This query is a traverse command which utilizes Edge and Vertice functions to list nodes and edges that are selected from a specific user. A nested select is used to achieve that goal.

*Result:* The result of the query in graph editor is shown in the figure bellow. The query Lists all the businesses that were reviewed by the users named “David”, using the limit clause for getting the first two-hundred (200) records indexed.


User-#25:5558

Properties

| | |
|---|---|
| @rid | #25:5558 |
| @class | User |
| elite | [] |
| compliments | {} |
| yelping_since | 2013-10 |
| user_id | JOFLsQy1gaqyUJoic7IOLg |
| average_stars | 1 |
| name | David |
| review_count | 1 |
| votes | {"cool":0,"useful":0,"funny":0} |

Graph Editor

```
traverse inE(), outV() from (select from user where name = 'David') limit 200
```

**Run:** Ctrl + Return | **Undo:** Ctrl/Cmd + Z | **Redo:** Ctrl/Cmd + Shift + Z
**Search :** Ctrl/Cmd + F | **Toggle Comment:** Ctrl/Cmd + / | **Autocomplete:** Ctrl + Space

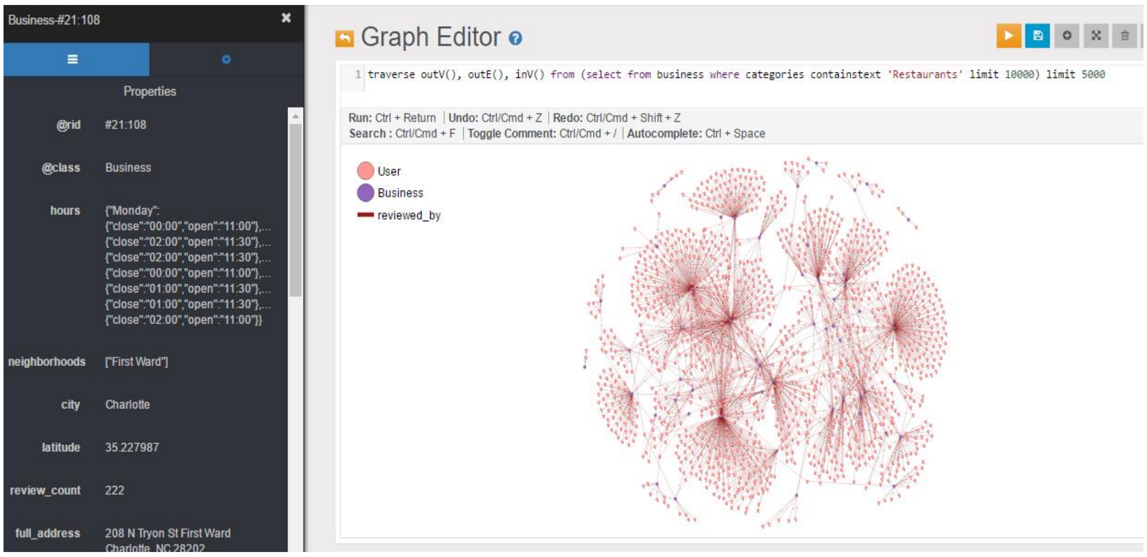
Figure 12: Query #6 Yelp database

***Query #7***

*SQL Syntax:* TRAVERSE outV(), outE(), inV() FROM (SELECT FROM business WHERE categories CONTAINSTEXT 'Restaurants' LIMIT 10000) LIMIT 5000

*Description:* This query is a traverse command which utilizes Edge and Vertice functions to list nodes and edges that are selected from a specific business. A nested select is used to achieve that goal.

*Result:* The result of the query in graph editor is shown in the figure bellow. Select the entire network of Restaurants and their reviewers limiting t the traverse records to five-thousands (5000).


Business-#21:108

Properties

| | |
|---|---|
| @rid | #21:108 |
| @class | Business |
| hours | {"Monday":<br>{"close":"00:00","open":"11:00"},...<br>{"close":"02:00","open":"11:30"},...<br>{"close":"02:00","open":"11:30"},...<br>{"close":"00:00","open":"11:00"},...<br>{"close":"01:00","open":"11:30"},...<br>{"close":"01:00","open":"11:30"},...<br>{"close":"02:00","open":"11:00"}} |
| neighborhoods | ["First Ward"] |
| city | Charlotte |
| latitude | 35.227987 |
| review_count | 222 |
| full_address | 208 N Tryon St First Ward<br>Charlotte, NC 28202 |

Graph Editor

```
traverse outV(), outE(), inV() from (select from business where categories containstext 'Restaurants' limit 10000) limit 5000
```

**Run:** Ctrl + Return | **Undo:** Ctrl/Cmd + Z | **Redo:** Ctrl/Cmd + Shift + Z
**Search :** Ctrl/Cmd + F | **Toggle Comment:** Ctrl/Cmd + / | **Autocomplete:** Ctrl + Space

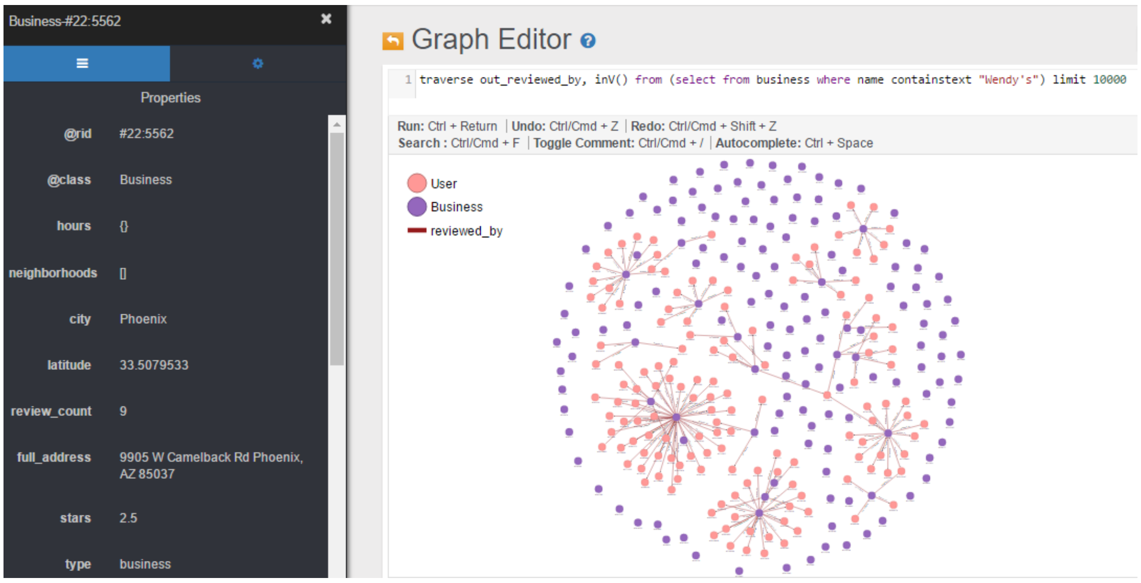
Figure 13: Query #7 Yelp database

***Query #8***

*SQL Syntax:* TRAVERSE out_reviewed_by, inV() FROM (SELECT FROM business WHERE name containstext "Wendy's") LIMIT 10000

*Description:* This query is a traverse command which utilizes Edge and Vertice functions to list nodes and edges that are selected from a specific business. A nested select is used to achieve that goal.

*Result:* The result of the query in graph editor is shown in the figure bellow. Retrieve reviews by users for all businesses named “Wendy’s” limiting the traversed records to ten-thousands (10000).


Business-#22:5562

Properties

| | |
|---|---|
| @rid | #22:5562 |
| @class | Business |
| hours | {} |
| neighborhoods | [] |
| city | Phoenix |
| latitude | 33.5079533 |
| review_count | 9 |
| full_address | 9905 W Camelback Rd Phoenix, AZ 85037 |
| stars | 2.5 |
| type | business |

Graph Editor

```
traverse out_reviewed_by, inV() from (select from business where name containstext "Wendy's") limit 10000
```

**Run:** Ctrl + Return | **Undo:** Ctrl/Cmd + Z | **Redo:** Ctrl/Cmd + Shift + Z
**Search :** Ctrl/Cmd + F | **Toggle Comment:** Ctrl/Cmd + / | **Autocomplete:** Ctrl + Space

Figure 14: Query #8 Yelp database

## 5.3 Experiment 2

Experiment 2 implements a graph representation of the Movies dataset in OrientDB using NetBeans for Java coding through OrientDB graph APIs. We used custom java classes to handle the Movies dataset files and create nodes and edges abiding with our proposed network schema. The main concept of work here is reading .csv files and then use graph API functions to populate the OrientDB database. Same as in first experiment, OrientDB provides interface for querying the database using SQL, as well as numerous commands for handling our data [60].

### 5.3.1 Movies Dataset

Description of the Movies dataset files is supplied at the Appendices. We used four (4) files as supplied by the grouplens site, the link.csv file for generating links, ratings.csv for ratings, tags.csv for tags and the movies.csv for movies description.

The datasets report 24,000,000 ratings, 27,000 movies, 27,000 links and 465,000 tags.

### 5.3.2 Movies Network Schema

The network schema for Experiment 2 is more complicated comparing with the first one (Yelp) we used, and we aimed on evaluating the graph complexity, depth of inheritance and creating nodes and edges from further manipulating the available dataset. In addition, the overall performance of OrientDB is tested as in the first experiment handling massive bulk of data.

In this experiment, Movie nodes, can have inbound edges Rate as well as outbound edges is_genre. Genre nodes have inbound edges is_genre while User nodes outbound edges Rate in a way that we form a relation like: Movies→is_genre→Genre, Users→Rate→Movies.

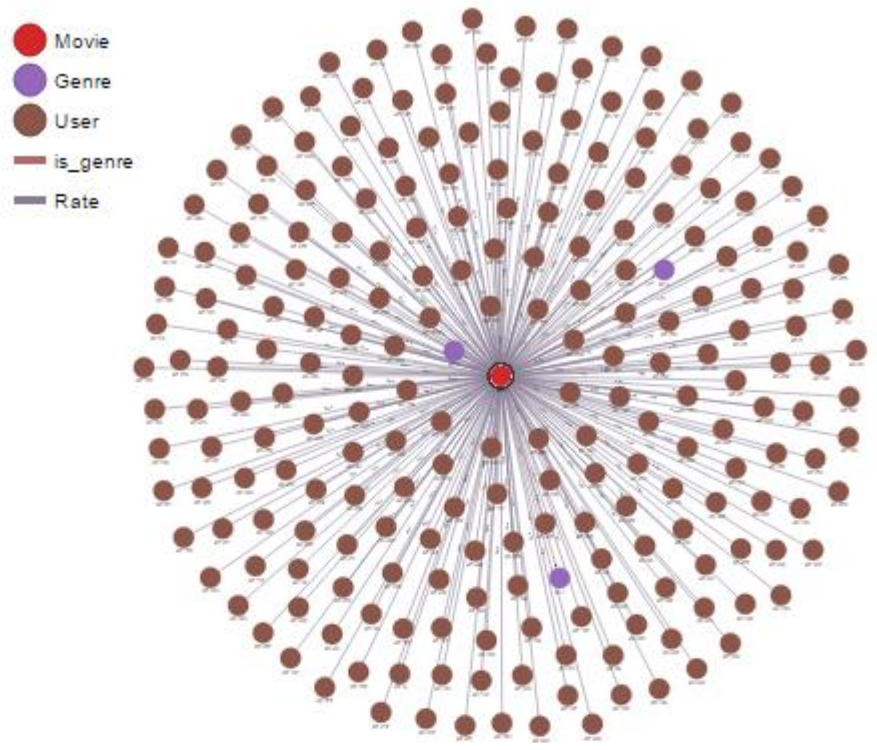


Figure 15: Network Schema Movies database

### 5.3.3 Implementation

The implementation was achieved using the OrientDB and the native Java graph API it offers. Appropriate packages where imported in a NetBeans project called Graph_Movies that achieves our main goal (database population with data) using four (4) Java classes.

These classes are Main.java, Config.java, Logger.java and Worker.java. Detailed functionality of each of these classes is supplied in the form of comments within the code accompanying this dissertation.

A general roadmap of the functionality of these classes is the following:

- Main.java creates a Graph instance that allows to connect with OrientDB, then requests back the existing transactional database and creates a Worker object that calls functions that generate nodes and edges forming our desired network schema.
- Config.java is a simple class that contains all the configuration settings required for OrientDB server. Examples are user, password etc.
- Logger.java is a default supplied class from the graph API that utilizes the user's Login to the Server.

All that mentioned, the network schema generated abides with the Star-schema network mentioned in Chapter 4.

According to the literature advised, star-schema is the most commonly used schema in social media, meaning that the whole implementation structure could be easily expanded or modified to allow the utilization of other real life datasets.

#### Querying Movies database

After compiling and running the java code produced in NetBeans, it is considered necessary to query the database to evaluate the produced product's performance, correctness and precision in results.

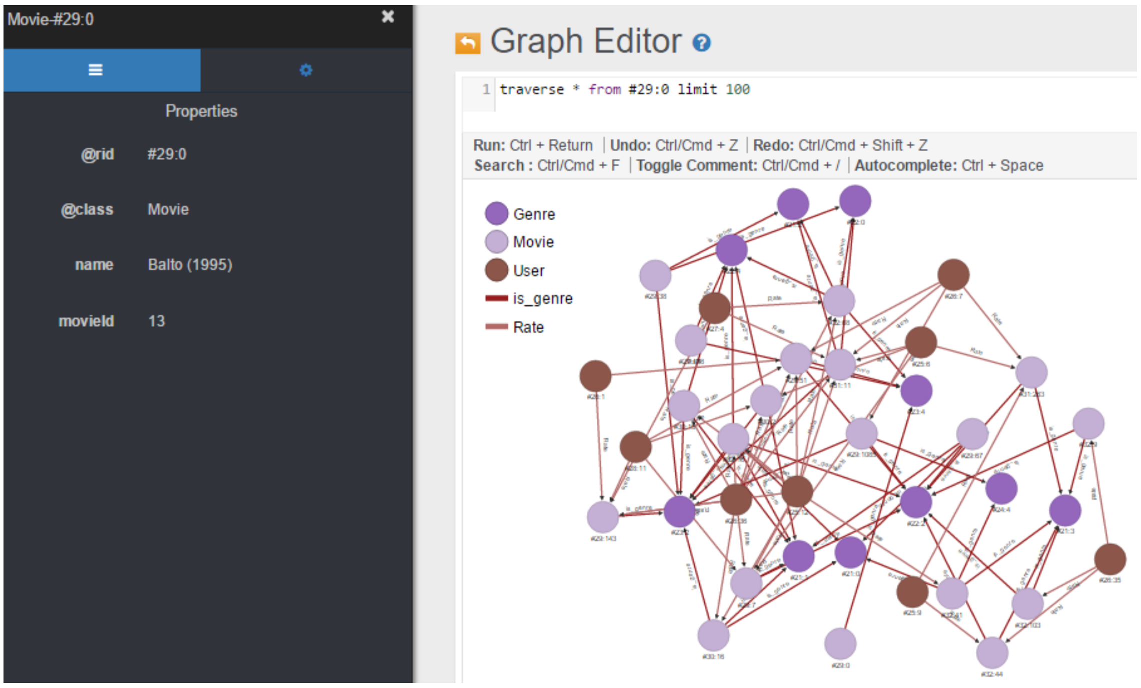
Some queries along with their results follow, that one could use to get useful information from a database utilizing a star-schema network.

***Query #1***
*SQL Syntax:* TRAVERSE * FROM #29:0 LIMIT 100

*Description:* The first query for this database is a rather simple traverse command which retrieves nodes and edges (relationships) across connected records that populate the database. #29:0 is the node record from where OrientDB will start traversing.

*Result:* The result of the query in graph editor is shown in the figure bellow. As we can observe each entry in the OrientDB has a unique index, for our example #29:0 which corresponds to the Movie named "Balto (1995)".

Another thing to mention, we are limiting the results produced to one hundred (100) using <limit 100>.



Movie-#29:0

Properties

| | |
|---|---|
| @rid | #29:0 |
| @class | Movie |
| name | Balto (1995) |
| movieId | 13 |

Graph Editor

```
traverse * from #29:0 limit 100
```

**Run:** Ctrl + Return | **Undo:** Ctrl/Cmd + Z | **Redo:** Ctrl/Cmd + Shift + Z
**Search :** Ctrl/Cmd + F | **Toggle Comment:** Ctrl/Cmd + / | **Autocomplete:** Ctrl + Space

Figure 16: Query #1 Movies database

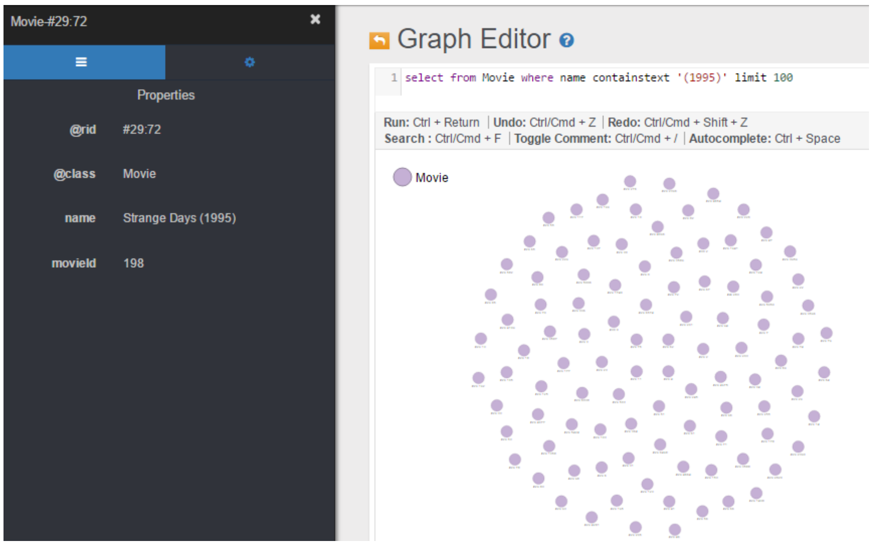
***Query #2***

*SQL Syntax:* SELECT FROM Movie WHERE name CONTAINSTEXT '(1995)' LIMIT 100

*Description:* The second query for this database is a rather simple select command which simply retrieves nodes that populate the database. It uses the containstext function to retrieve all the movies that their name field contains "1995".

*Result:* The result of the query in graph editor is shown in the figure bellow. This query retrieves up to 100 Movies that were released in 1995.

Again, we are limiting the results produced to one hundred (100) using <limit 100>.


Movie-#29:72

Properties

| | |
|---|---|
| @rid | #29:72 |
| @class | Movie |
| name | Strange Days (1995) |
| movieId | 198 |

Graph Editor

```
select from Movie where name containstext '(1995)' limit 100
```

**Run:** Ctrl + Return | **Undo:** Ctrl/Cmd + Z | **Redo:** Ctrl/Cmd + Shift + Z
**Search :** Ctrl/Cmd + F | **Toggle Comment:** Ctrl/Cmd + / | **Autocomplete:** Ctrl + Space

Figure 17: Query #2 Movies database

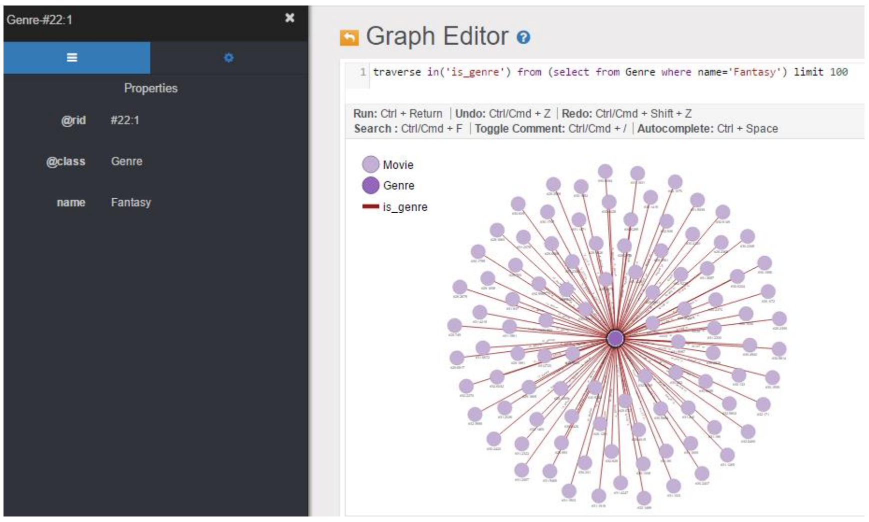
***Query #3***

*SQL Syntax:* TRAVERSE in('is_genre') FROM (SELECT FROM Genre WHERE name = 'Fantasy') LIMIT 100

*Description:* The third query for this database is a traverse command which retrieved fantasy movies from the database. To achieve that it calls the in (<edge class>) function as well an nested query.

*Result:* The result of the query in graph editor is shown in the figure bellow. This query shows all Movies that are fantasy movies.

We are limiting the results produced to one-hundred (100) fantasy movies, using <limit 100>.


Genre-#22:1

Properties

| | |
|---|---|
| @rid | #22:1 |
| @class | Genre |
| name | Fantasy |

Graph Editor

```
traverse in('is_genre') from (select from Genre where name='Fantasy') limit 100
```

**Run:** Ctrl + Return | **Undo:** Ctrl/Cmd + Z | **Redo:** Ctrl/Cmd + Shift + Z
**Search :** Ctrl/Cmd + F | **Toggle Comment:** Ctrl/Cmd + / | **Autocomplete:** Ctrl + Space

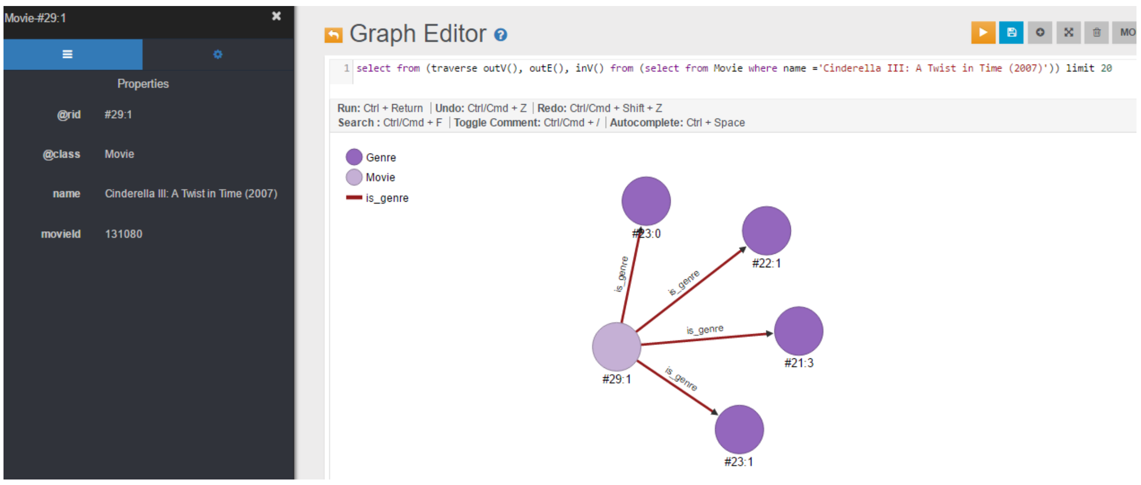
Figure 18: Query #3 Movies database

***Query #4***

*SQL Syntax:* SELECT FROM (TRAVERSE outV(), outE(), inV() FROM (SELECT FROM Movie WHERE name ='Cinderella III: A Twist in Time (2007)')) LIMIT 20

*Description:* The fourth query for this database is a select command which uses a nested traverse and nested select. It used a plethora of functions (eg outV(), outE()) to retrieve the requested results.

*Result:* The result of the query in graph editor is shown in the figure bellow. This query traverses each edge "is_genre" that is connected to the movie "Cinderella III" getting the genres for this movie. We are limiting the results produced to twenty (20), using <limit 20>.



| | |
|---|---|
| @rid | #29:1 |
| @class | Movie |
| name | Cinderella III: A Twist in Time (2007) |
| movieId | 131080 |

```
select from (traverse outV(), outE(), inV() from (select from Movie where name ='Cinderella III: A Twist in Time (2007)')) limit 20
```

Run: Ctrl + Return | Undo: Ctrl/Cmd + Z | Redo: Ctrl/Cmd + Shift + Z
Search : Ctrl/Cmd + F | Toggle Comment: Ctrl/Cmd + / | Autocomplete: Ctrl + Space

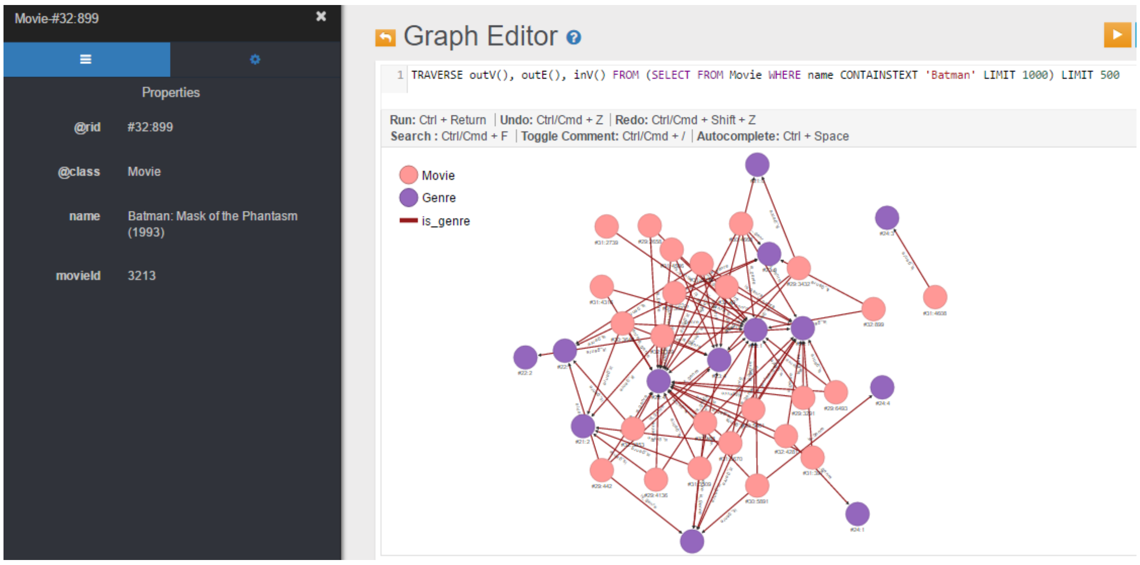
Figure 19: Query #4 Movies database

***Query #5***

*SQL Syntax:* TRAVERSE outV(), outE(), inV() FROM (SELECT FROM Movie WHERE name CONTAINSTEXT 'Batman' LIMIT 1000) LIMIT 500

*Description:* This query is a traverse command which utilizes Edge and Vertice functions to list nodes and edges that are selected from a specific Movies. A nested select is used to achieve that goal.

*Result:* The result of the query in graph editor is shown in the figure bellow. Select the entire network of Batman Movies and their reviewers limiting the traverse records to five-hundred (500).


Movie-#32:899

Properties

| | |
|---|---|
| @rid | #32:899 |
| @class | Movie |
| name | Batman: Mask of the Phantasm (1993) |
| movieId | 3213 |

```
TRAVERSE outV(), outE(), inV() FROM (SELECT FROM Movie WHERE name CONTAINSTEXT 'Batman' LIMIT 1000) LIMIT 500
```

Run: Ctrl + Return | Undo: Ctrl/Cmd + Z | Redo: Ctrl/Cmd + Shift + Z
Search : Ctrl/Cmd + F | Toggle Comment: Ctrl/Cmd + / | Autocomplete: Ctrl + Space

Figure 20: Query #5 Movies database

## Calculating Graph measures

After performing some queries to the produced database, we will be using Gephi plugins for OrientDB to get the most common statistic measures regarding our graph. We will use a fixed amount of entries to produce tables for comparison for our examples instead of using the whole dataset entries.

According to Gephi documentation, queries are transformed in a “http get” form allowing to withdraw entries from OrientDB studio. Thus, in this section we will perform three (3) such queries, each for a different proportion of the database entries, noting down the results. The queries could not run for 100% of the database due to lack of physical memory.

### *Query #1*

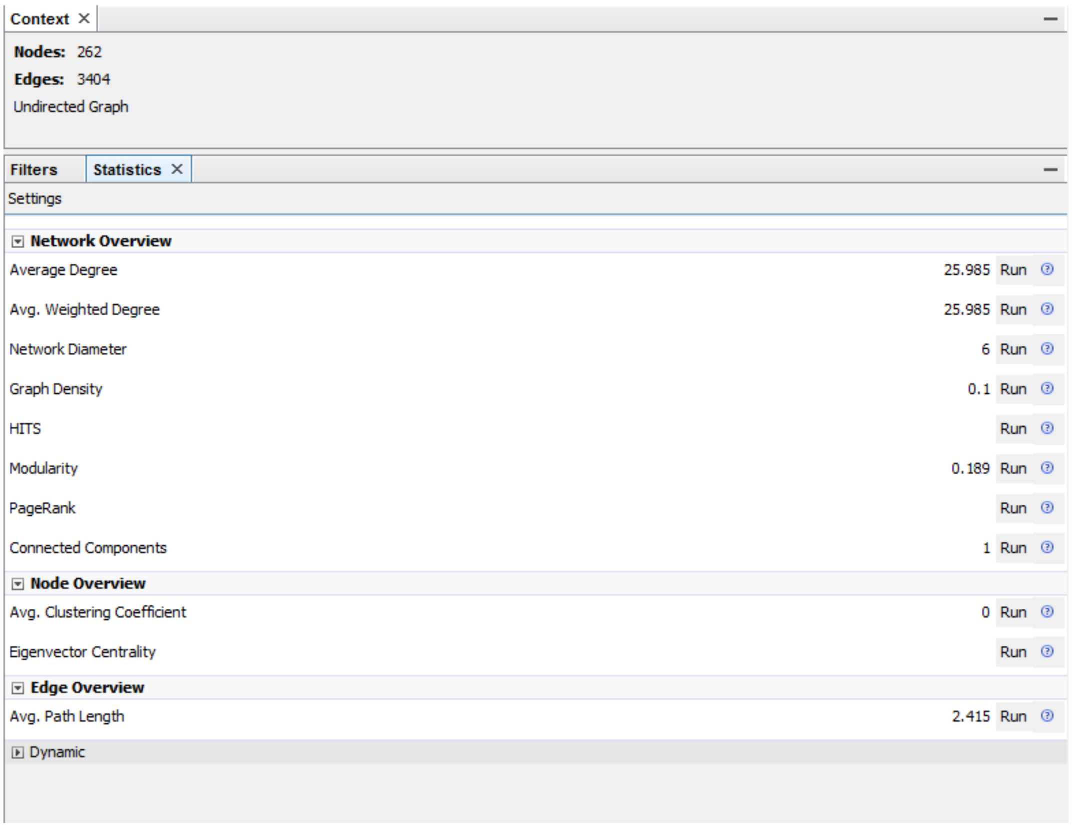
*Syntax:*

http://localhost:2480/gephi/Graph_Movies/sql/traverse%20*%20from%20Movie%20/1000

*Description:* It traverses everything (Edges, Vertices) from Vertice Movies limiting the results to one-thousand (1000).

*Result:*

The execution of the query returns some measures for Movies database. Figure 21 shows the results regarding the *Network Overview*.


Context

**Nodes:** 262
**Edges:** 3404
Undirected Graph

Filters | Statistics

Settings

**Network Overview**

```
Average Degree                 25.985  Run
Avg. Weighted Degree           25.985  Run
Network Diameter                    6  Run
Graph Density                     0.1  Run
HITS                                   Run
Modularity                      0.189  Run
PageRank                               Run
Connected Components                1  Run
```

**Node Overview**

```
Avg. Clustering Coefficient         0  Run
Eigenvector Centrality                 Run
```

**Edge Overview**

```
Avg. Path Length                2.415  Run
```

Dynamic

Figure 21: Query #1 Gephi overview

*Degree Distribution* is a measure that returns the average number edges for a node.

Average Weighted Degree: 25.985

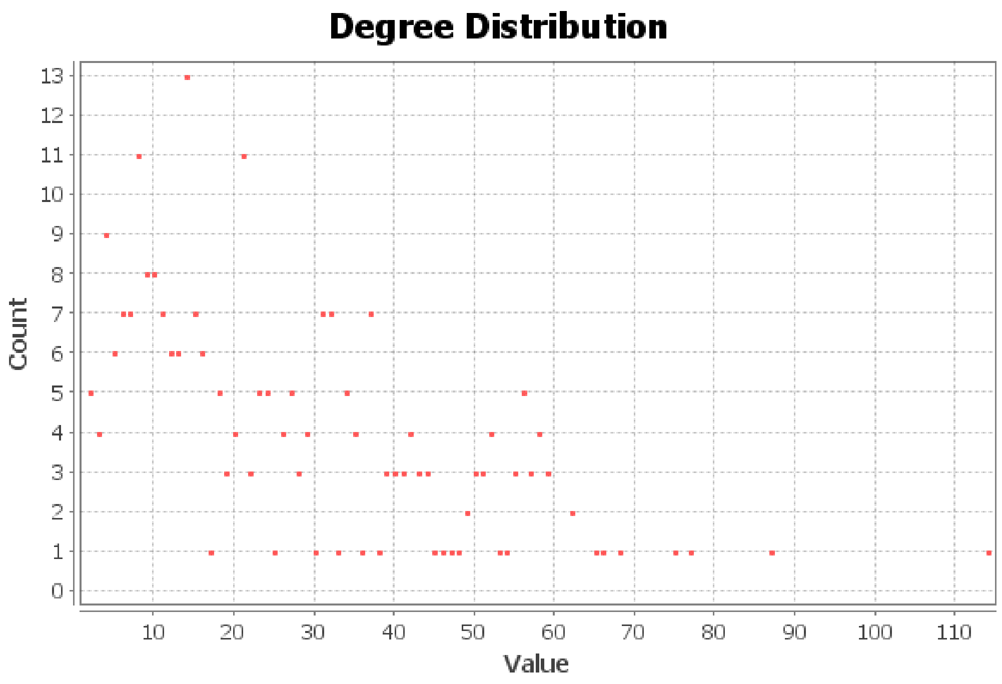


Figure 22: Query #1 Avg. Weighted Degree

*Graph Density* is a measure that shows how close is the generated graph to being complete. A graph is marked as complete when every pair of distinct nodes connects with a unique edge.

The following measures are implemented as proposed in [94]. Short descriptions follow:

*Distance* is the average graph-distance between all pairs of nodes. Connected nodes have distance value 1.

*Diameter or Network Diameter* is the highest graph distance between two nodes within the network.

*Betweenness Centrality* calculates how often each node appears on shortest paths between nodes in the network.

*Harmonic Closeness Centrality* calculates the average distance from a specific node to all other nodes within the network.

*Eccentricity distribution* calculates the distance from a specific node to the farthest node from it within the network.

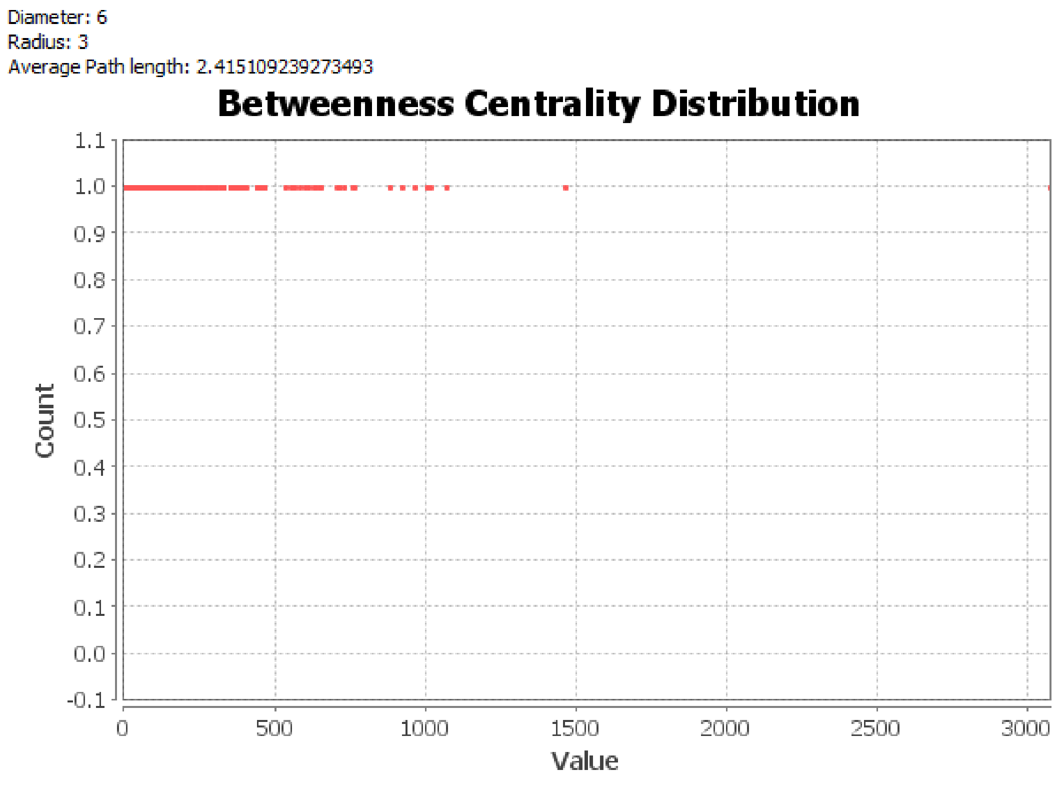

Diameter: 6
Radius: 3
Average Path length: 2.415109239273493

Figure 23: Query #1 Betweenness Centrality Distribution

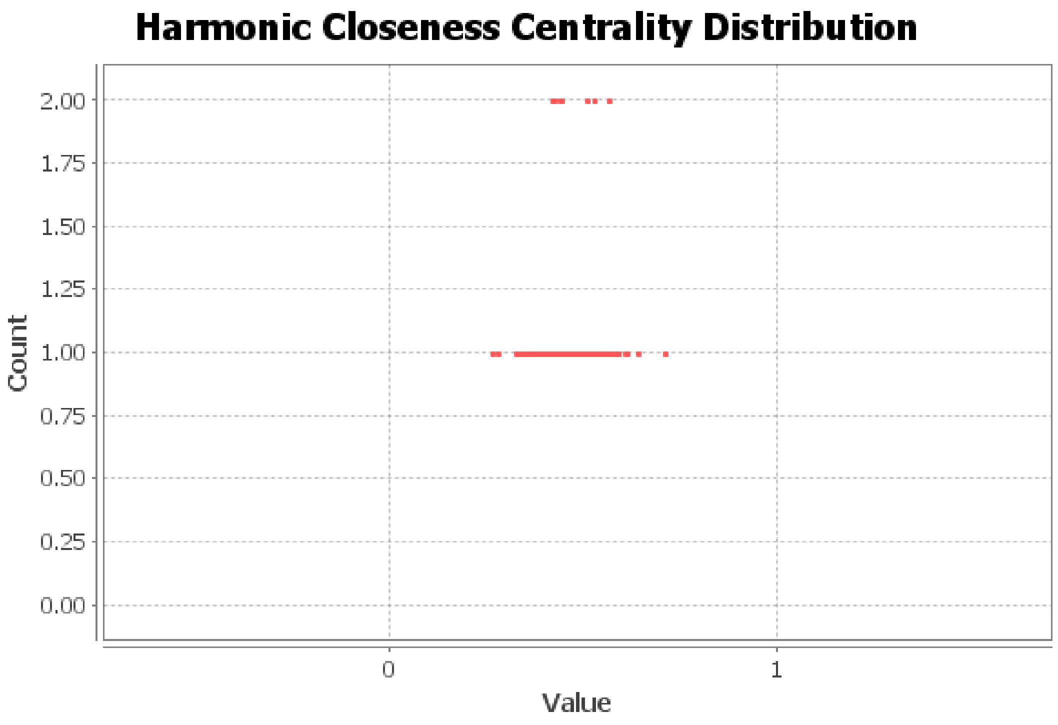


Figure 24: Query #1 Harmonic Closeness Centrality Distribution

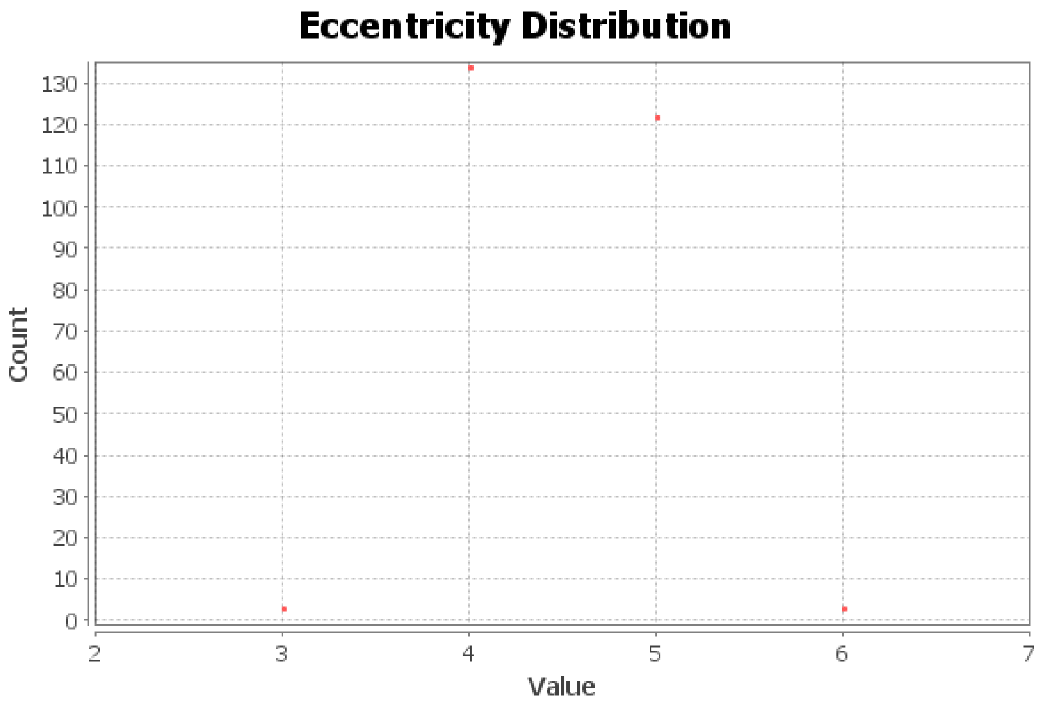
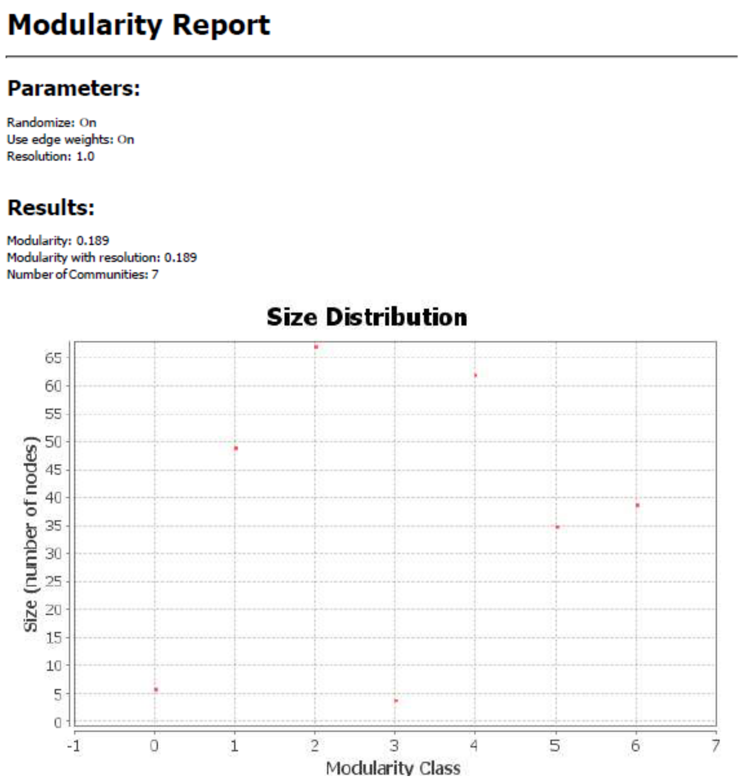


Figure 25: Query #1 Eccentricity Distribution

*Modularity* is calculated as proposed in [95]. Modularity is a measure that shows how well an information network can decompose into modular communities.



## Modularity Report

**Parameters:**

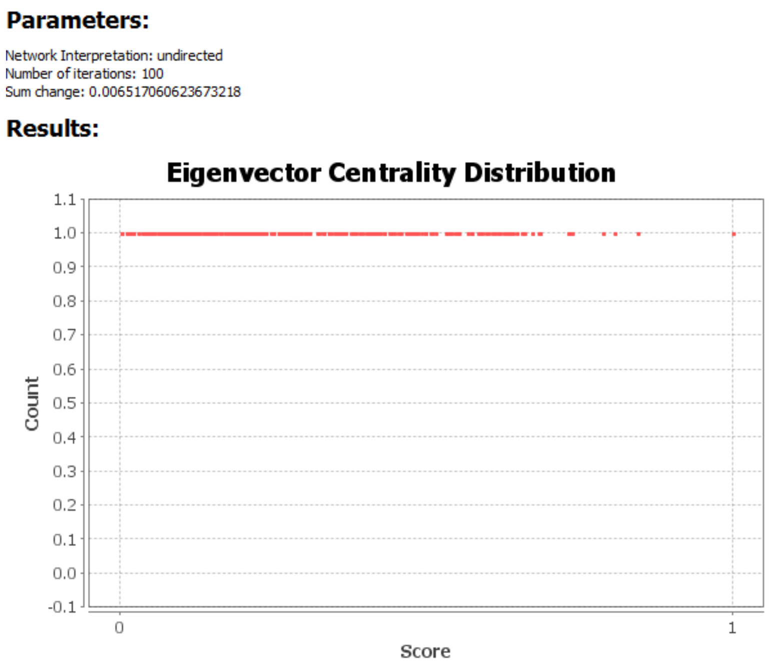
Randomize: On
Use edge weights: On
Resolution: 1.0

**Results:**

Modularity: 0.189
Modularity with resolution: 0.189
Number of Communities: 7

Figure 26: Query #1 Modularity

*Eigenvector* is a measure that finds that importance of a node within the information network based on how many connections (links) this node has.



**Parameters:**

Network Interpretation: undirected
Number of iterations: 100
Sum change: 0.006517060623673218

**Results:**

Figure 27: Query #1 Eigenvector Centrality

***Query #2***

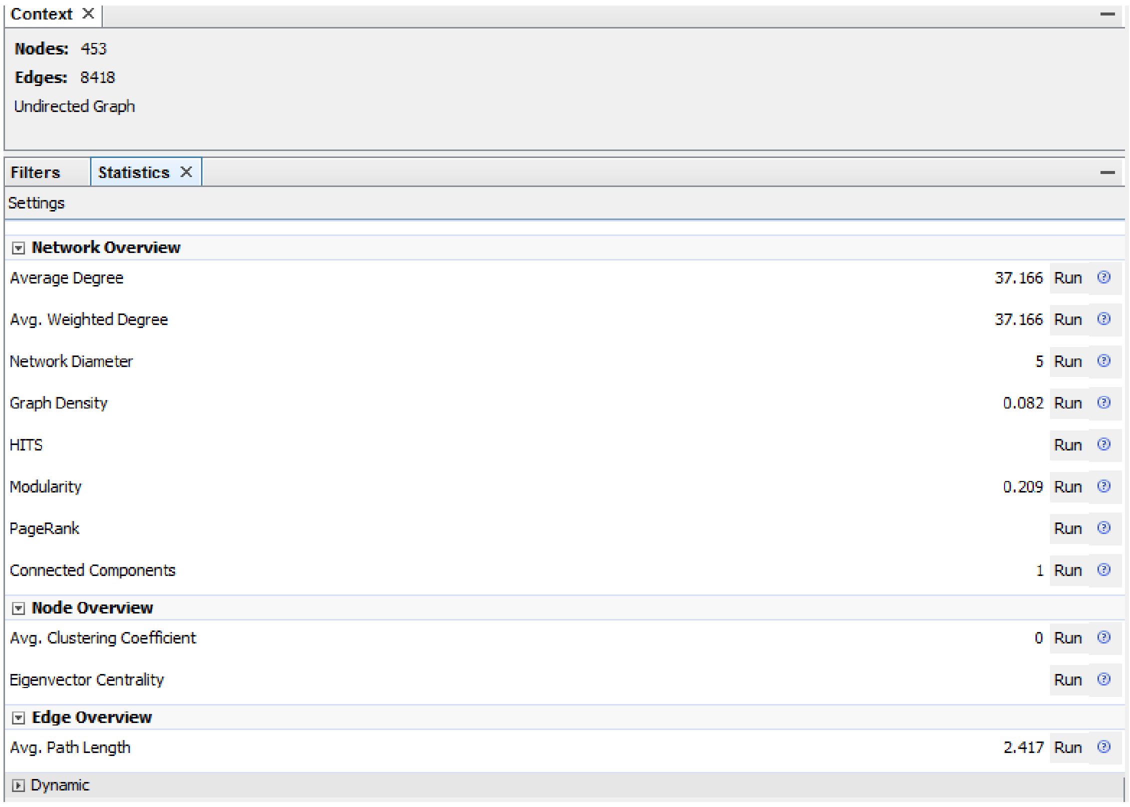
*Syntax:*

http://localhost:2480/gephi/Graph_Movies/sql/traverse%20*%20from%20Movie%20/2000

*Description:* It traverses everything (Edges, Vertices) from Vertice Movies limiting the results to two-thousand (2000).

*Result:*

The procedure followed for Query #1 will be repeated on this results section.


Context

**Nodes:** 453
**Edges:** 8418
Undirected Graph

Filters | Statistics

Settings

**Network Overview**

| | | | |
|---|---|---|---|
| Average Degree | 37.166 | Run | |
| Avg. Weighted Degree | 37.166 | Run | |
| Network Diameter | 5 | Run | |
| Graph Density | 0.082 | Run | |
| HITS | | Run | |
| Modularity | 0.209 | Run | |
| PageRank | | Run | |
| Connected Components | 1 | Run | |

**Node Overview**

| | | | |
|---|---|---|---|
| Avg. Clustering Coefficient | 0 | Run | |
| Eigenvector Centrality | | Run | |

**Edge Overview**

| | | | |
|---|---|---|---|
| Avg. Path Length | 2.417 | Run | |

Dynamic

Figure 28: Query #2 Gephi overview

Average Weighted Degree: 37.166

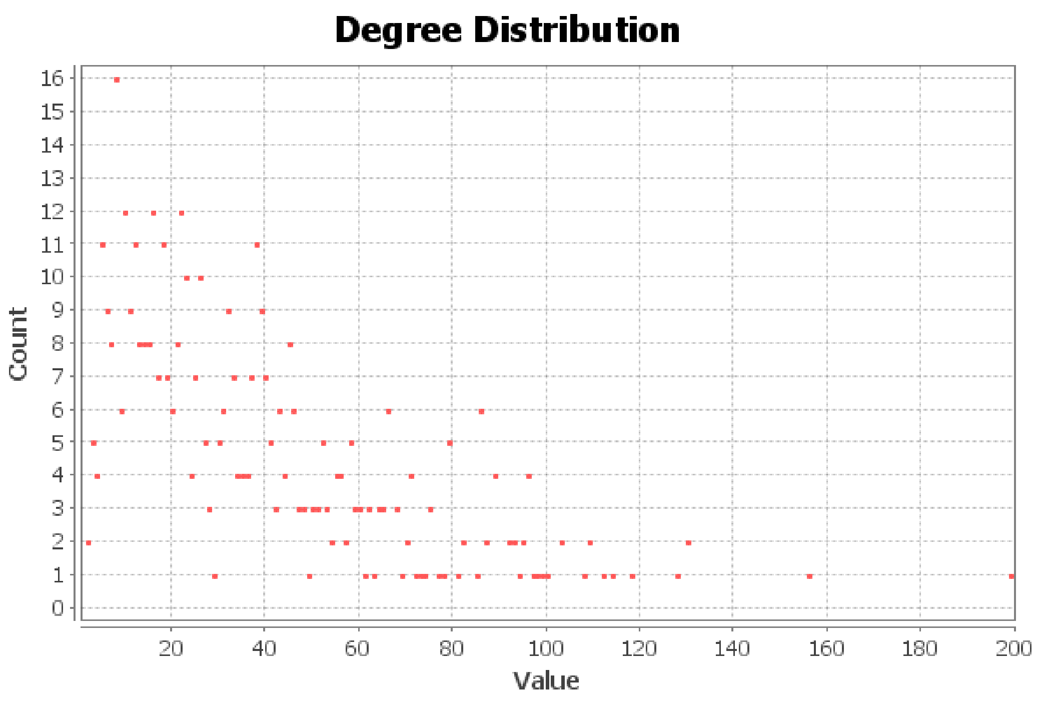


Figure 29: Query #2 Avg. Weighted Degree

Diameter: 5
Radius: 3
Average Path length: 2.4166324796342966

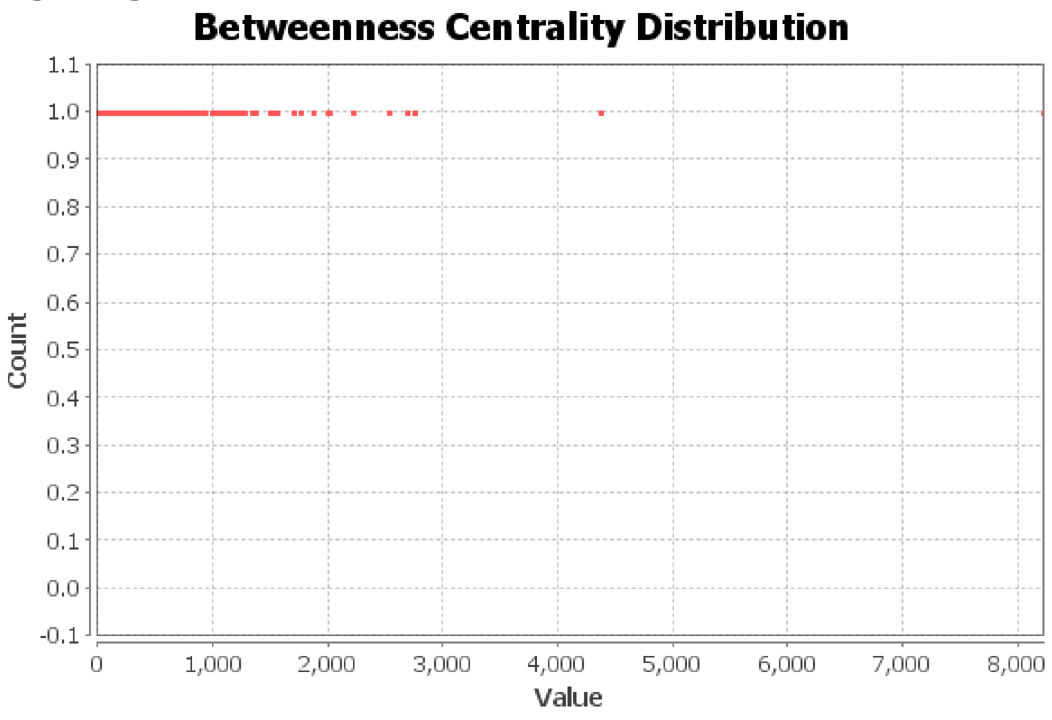


Figure 30: Query #2 Betweenness Centrality Distribution

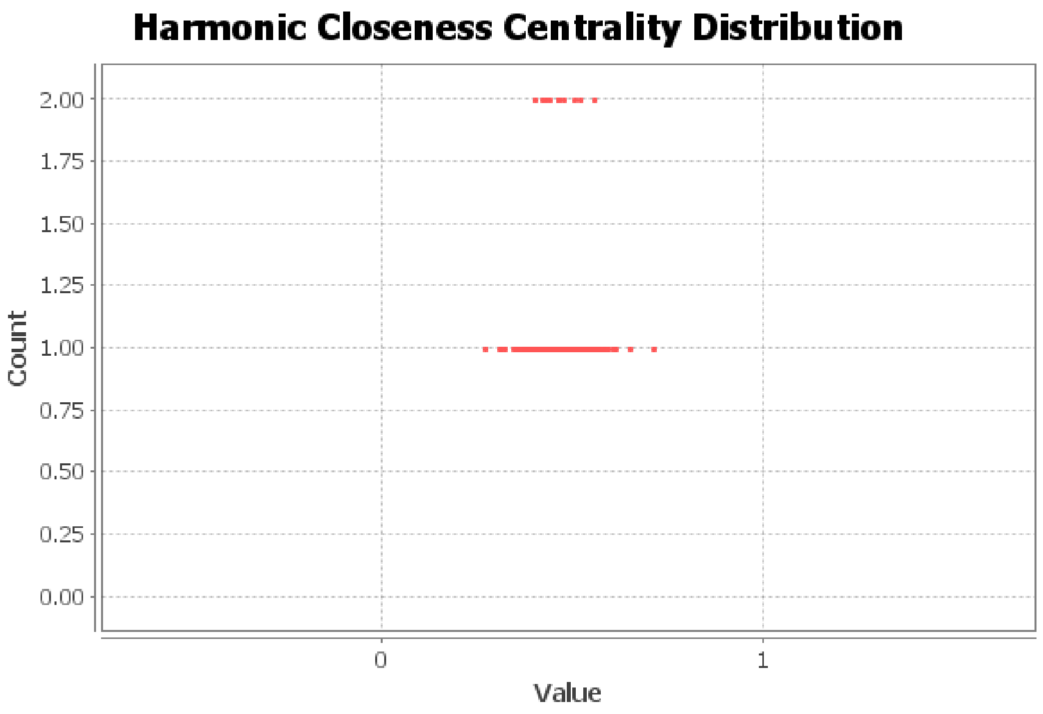


Figure 31: Query #2 Harmonic Closeness Centrality Distribution

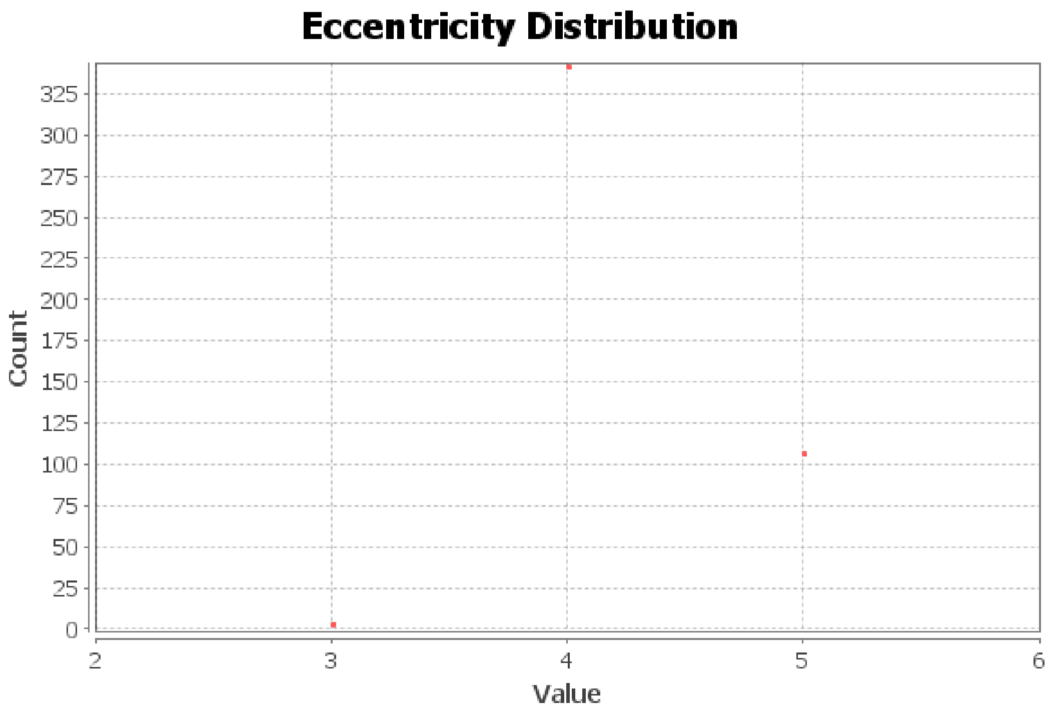


Figure 32: Query #2 Eccentricity Distribution

## Modularity Report

### Parameters:

Randomize: On
Use edge weights: On
Resolution: 1.0

### Results:

Modularity: 0.209
Modularity with resolution: 0.209
Number of Communities: 5

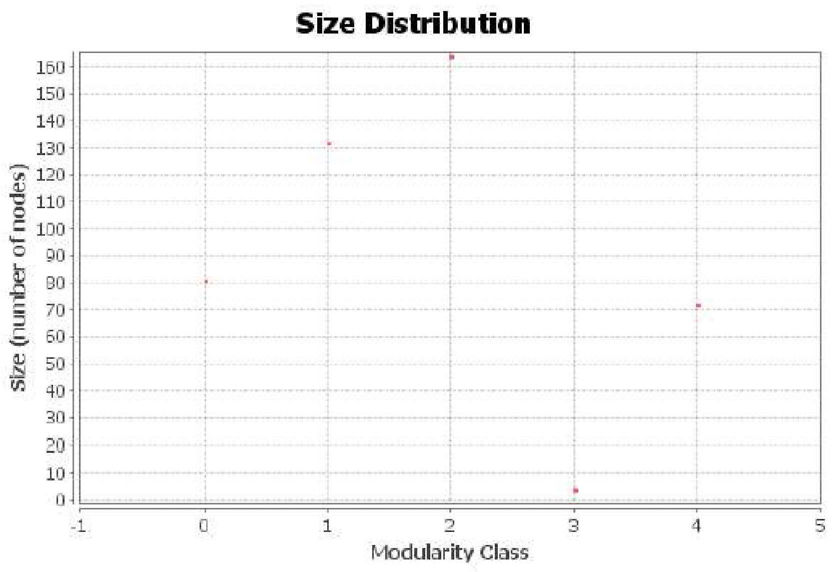


Figure 33: Query #2 Modularity

### Parameters:

Network Interpretation: undirected
Number of iterations: 100
Sum change: 0.010588772703639691

### Results:

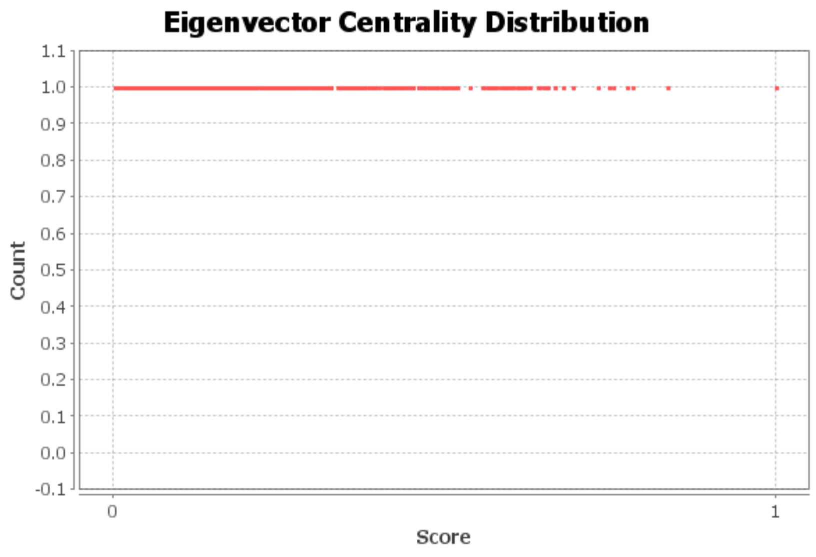


Figure 34: Query #2 Eigenvector Centrality

***Query #3***

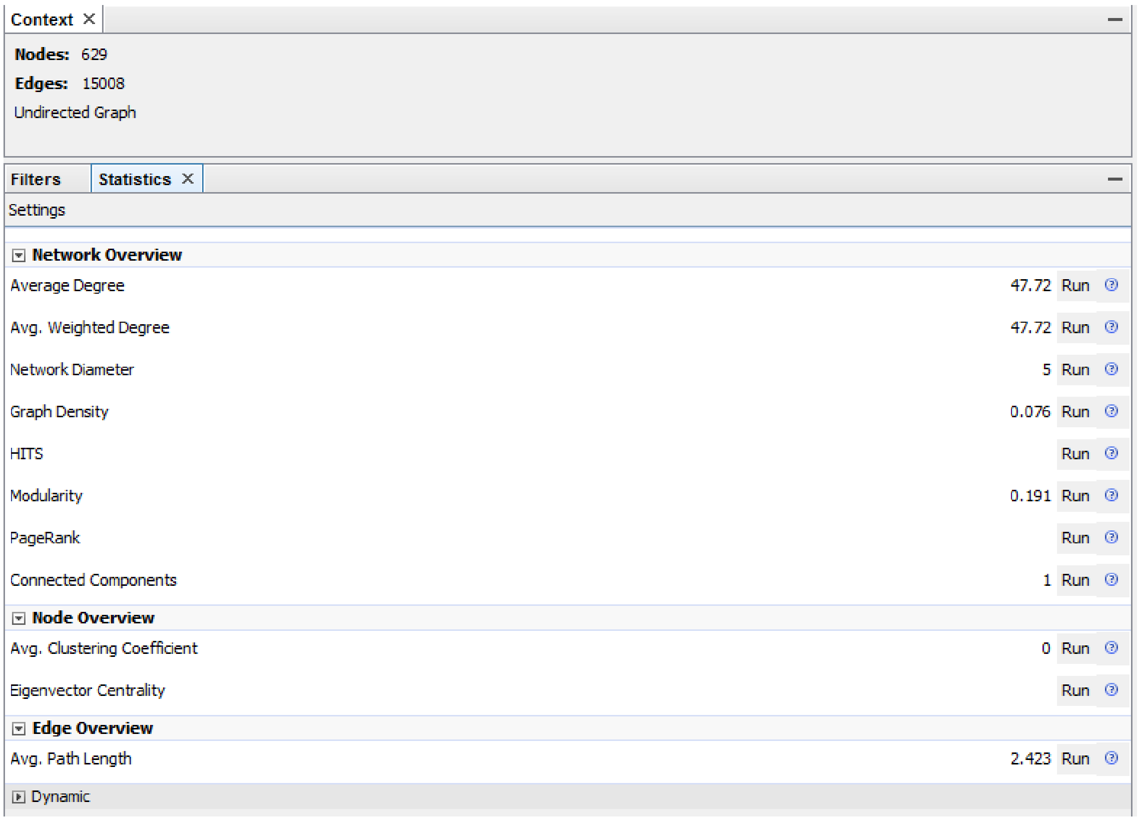
*Syntax:*

http://localhost:2480/gephi/Graph_Movies/sql/traverse%20*%20from%20Movie%20/3000

*Description:* It traverses everything (Edges, Vertices) from Vertice Movies limiting the results to three-thousand (3000).

*Result:*

The procedure followed for Query #2, and Query #3 is repeated on this results section.


Context

**Nodes:** 629
**Edges:** 15008
Undirected Graph

Filters | Statistics

Settings

**Network Overview**

| | | |
|---|---|---|
| Average Degree | 47.72 | Run |
| Avg. Weighted Degree | 47.72 | Run |
| Network Diameter | 5 | Run |
| Graph Density | 0.076 | Run |
| HITS | | Run |
| Modularity | 0.191 | Run |
| PageRank | | Run |
| Connected Components | 1 | Run |

**Node Overview**

| | | |
|---|---|---|
| Avg. Clustering Coefficient | 0 | Run |
| Eigenvector Centrality | | Run |

**Edge Overview**

| | | |
|---|---|---|
| Avg. Path Length | 2.423 | Run |

Dynamic

Figure 35: Query #3 Gephi overview

Average Weighted Degree: 47.720

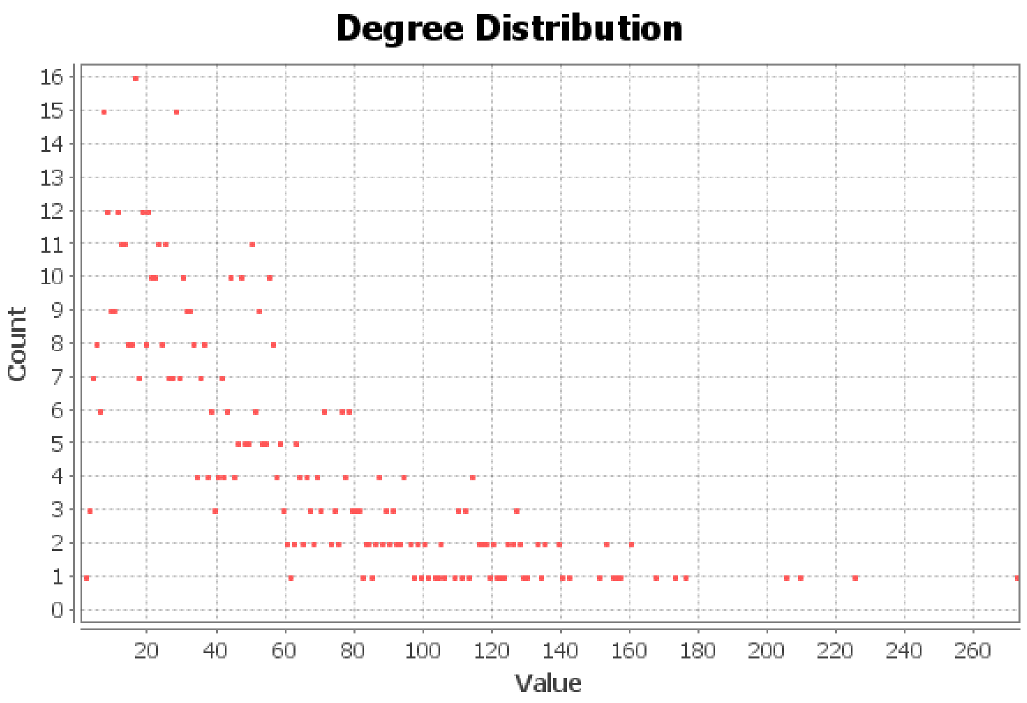


Figure 36: Query #3 Avg. Weighted Degree

Diameter: 5
Radius: 3
Average Path length: 2.4231061334845525

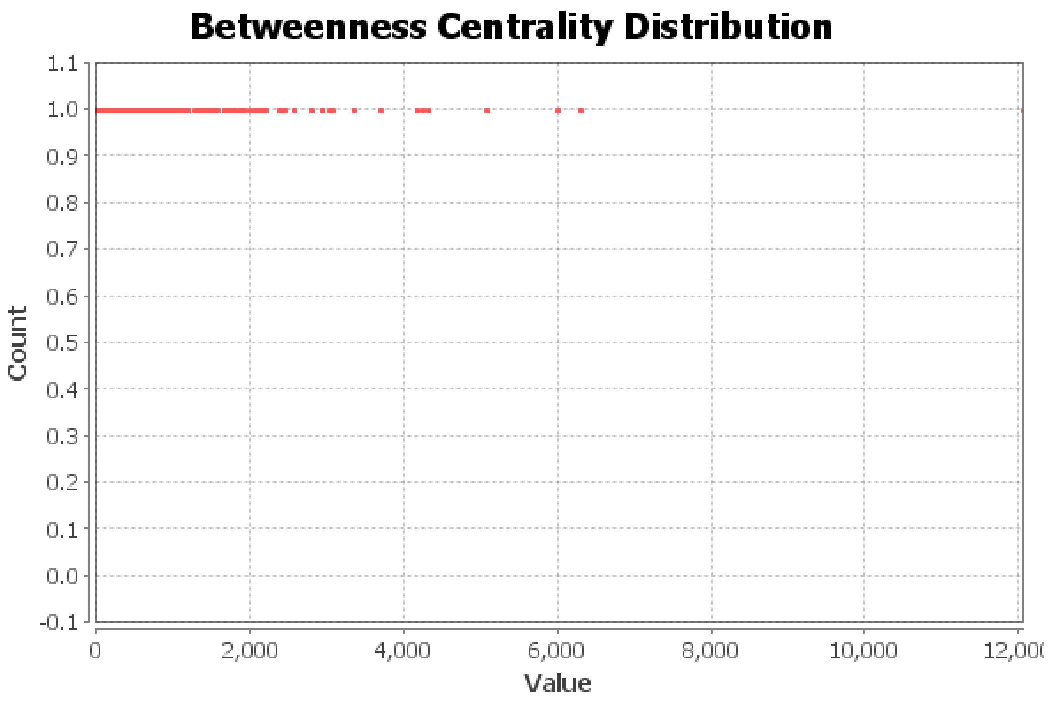


Figure 37: Query #3 Betweenness Centrality Distribution

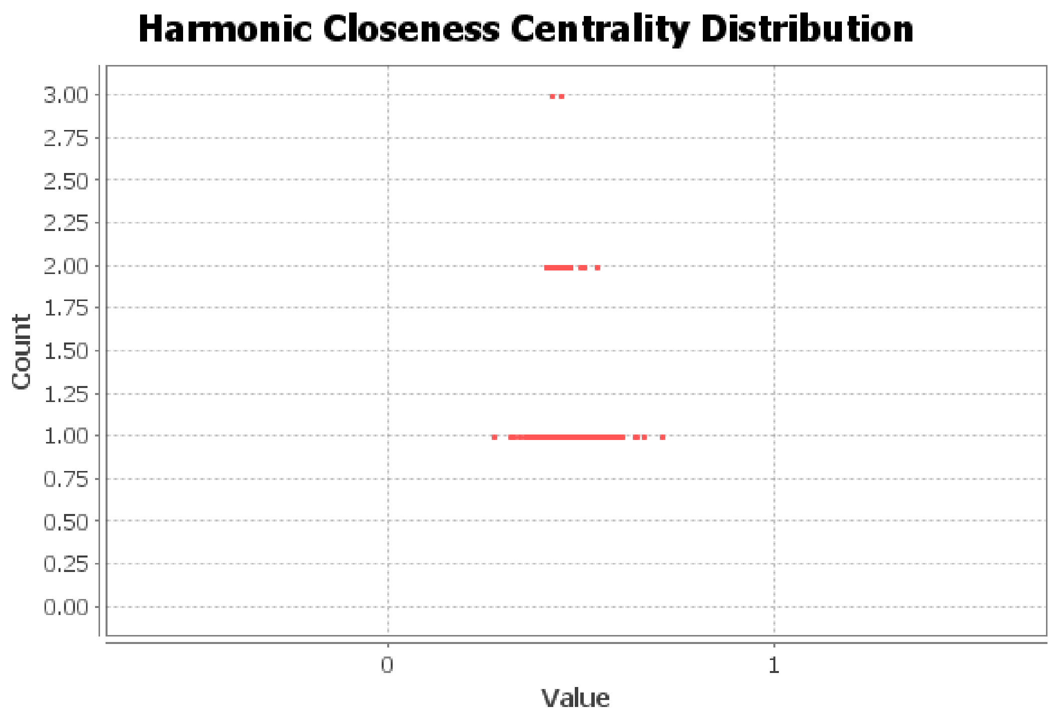


Figure 38: Query #3 Harmonic Closeness Centrality Distribution

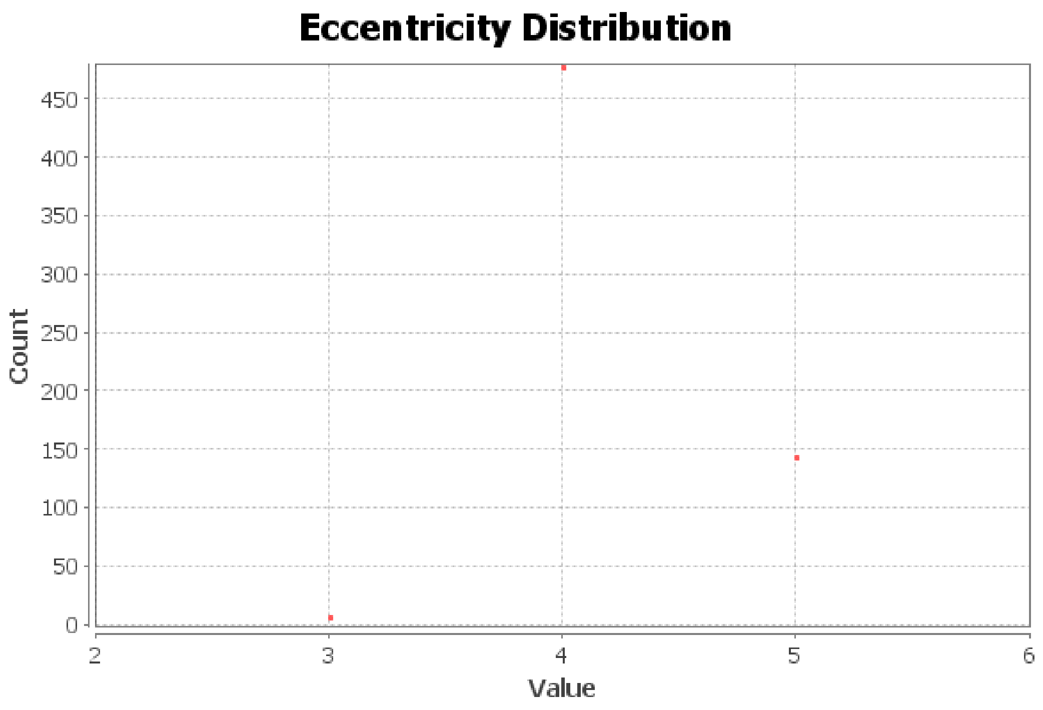


Figure 39: Query #3 Eccentricity Distribution

## Modularity Report

### Parameters:

Randomize: On
Use edge weights: On
Resolution: 1.0

### Results:

Modularity: 0.191
Modularity with resolution: 0.191
Number of Communities: 6

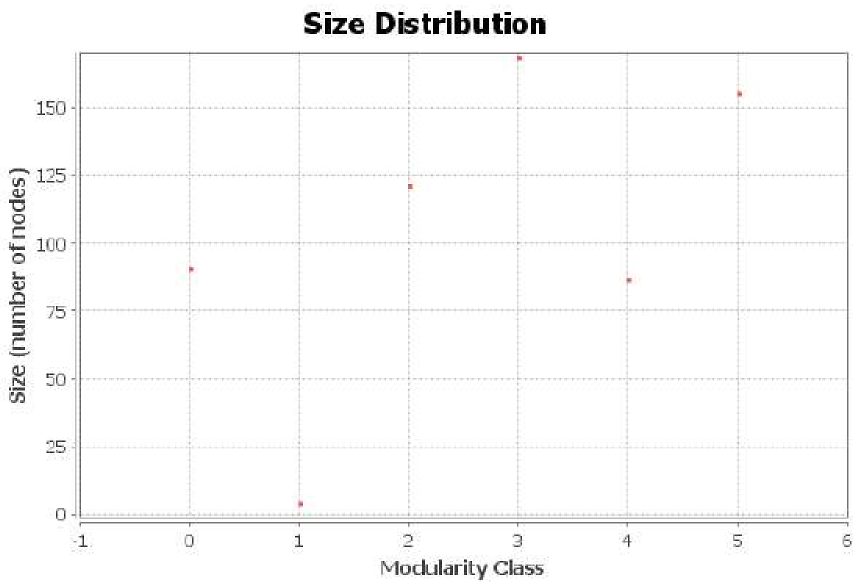


Figure 40: Query #3 Modularity

### Parameters:

etwork Interpretation: undirected
umber of iterations: 100
um change: 0.012620767925334094

### Results:

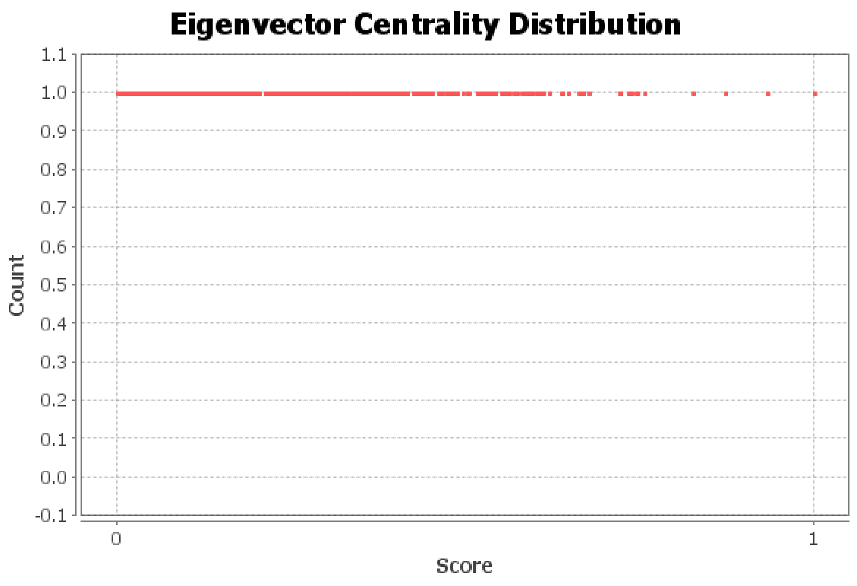


Figure 41: Query #3 Eigenvector Centrality

***Results discussion***

Using Gephi we can retrieve measures for generated graphs as discussed in Tools chapter. Measures calculated for the Movies dataset have a generic meaning as the input nodes and edges are the result of a query requesting a proportion of the Movies' database entries and their connections.

Our aim is to show the methodology followed; combining OrientDB with Gephi to generate comparable measures for our results.

For example, Table 6, shows that when comparing network diameter values for Query#1 and Query#3 we observe the when we calculate this metric inputting more entries (nodes and edges) the network diameter value falls to 5, meaning that the highest graph distance between two nodes within the network is reduced.

Also, for Query#1 and Query#2 communities observed from the output of the measure falls from 7 to 5 respectively.

The prospects from graph measures in complex information networks are numerous, as discussed in the theoretical part of this dissertation.

Next, two (2) tables follow summarizing the values of measures withdrawn from Gephi. Definitions for each of the measures in Table 6 are given in Query #1 of this section.

Table 5: Gephi Queries entity population

| Query | Nodes | Edges |
|---|---|---|
| **Query #1** | 262 | 3404 |
| **Query #2** | 453 | 8418 |
| **Query #3** | 629 | 15008 |

Table 6: Overview of results Gephi measures

| Measure | Query #1 | Query #2 | Query #3 |
|---|---|---|---|
| **Avg. (Weighted) Degree** | 25.986 | 37.166 | 47.720 |
| **Network Diameter** | 6 | 5 | 5 |
| **Graph Density** | 0.1 | 0.082 | 0.076 |
| **Modularity** | 0.189 | 0.209 | 0.191 |
| **Communities** | 7 | 5 | 6 |
| **Eigenvector (Sum change)** | 0.0065170 | 0.0105887 | 0.0126207 |
| **Average path length** | 2.515 | 2.416 | 2.423 |

## 5.4 Experiment 3

Experiment 3 represents an implementation of the RankClus algorithm as proposed in [37], using a synthetic dataset called PROFDB in NetBeans with Java. We used custom Java classes for handling the PROFDB dataset to link nodes abiding with our proposed network schema. The main concept of work here is; reading an input xml file and then using Java functions to implement the RankClus algorithm.

This experiment is different from the other two (2) experiments conducted, as we do not aim on populating a DBMS but to test the functionality of an algorithm. The output of this experiment produces clusters that represent MSc programs, with each cluster containing ranked professors based on the frequency and type of the courses they teach.

### 5.4.1 IHU Professors-MSc Programs (PROFDB) Dataset

A detailed description of the PROFDB dataset files is supplied at the Appendices. There is a single .xml file containing the data we will be using for running the java code.

The dataset reports 10 Professors, 4 MSc programs, 20 courses, 40 links Professors-courses and 20 links courses-MSc programs. The connections between Professors-courses were randomly generated as well as the courses-MSc programs, meaning than the dataset does not represent real world connections.

### 5.4.2 RankClus

A summarization of the functionality of RankClus algorithm follows which was implemented in Java. A detailed reference of this algorithm's functionality is found in [37].

Generally, RankClus converts each input object considering a mix model of the current cluster and then calculates the new positions of the objects into the nearest clusters applying the new attributes. The above process is repeated until a point that each new iteration does not update the new position significantly.

Clustering improves with each consequent iteration since identical objects with new attributes will be appointed to the same group.

Ranking improves generating better attributes for the next iterations, thus, also improving clustering.

The algorithm functions in five (5) steps as described below:

*Step 1*: The generation of the initial clusters. Each of the dataset objects get assigned with a cluster label in a random manner.

*Step 2:* The ranking for each cluster starts. Calculate conditional rank for generated clusters of step 1. Also, check if any clusters are empty, if empty restart.

*Step 3:* Calculate the mixture model component coefficients. Start an iteration (empirically set to 5) for getting the new values for objects and center points of each cluster.

*Step 4:* Recalculate clusters. Get the distance of all objects with the center cluster they belong and move them to the nearest cluster.

*Step 5:* Steps 2,3,4 are repeated until clusters change insignificantly after each iteration or a fixed amount of iterations (it is a predefined number we set as argument to the algorithm).

### 5.4.3 Implementation

The implementation was achieved using NetBeans. Appropriate packages where imported in a NetBeans project called rankclus that achieves our main goal (read/handle data) using nice (9) Java classes.

These classes are Cluster.java, Edge.java, Parser.java, Rank.java, RankClus.java (main class), RankProfessor.java, SingleCluster.java, Store.java and Vertex.java. Detailed functionality of each of these classes is supplied in the form of comments within the code accompanying this dissertation.

A general roadmap of the functionality of these classes is the following:

- Cluster.java creates a Cluster instance that performs the clustering. It includes functions that utilize steps 2, 3 and 4 of the RankClus algorithm and initiates the inner-iterations.
- Edge.java is a simple class that contains the constructor for an Edge object and its properties e.g. id, source, destination.
- Parser.java is a class that handles xml parsing of the dataset.
- Rank.java handles the ranking of the professors as well as the comparison that we need to execute during the inner steps of the algorithm.
- RankClus.java is the main class of the project, it handles the creation of the required objects as well as the call of the functions to execute the RankClus algorithm, printing step-by-step proper execution messages.
- RankProfessor.java simply contains a constructor for a RankProfessor object.
- SingleCluster.java initiates matrices and constructs a SingleCluster object.

- Store.java contains functions for storing and handling person, program and course data entities, to HashMaps.
- Vertex.java is a simple class that contains the constructor for an Vertex object and its properties e.g. id, name, type.

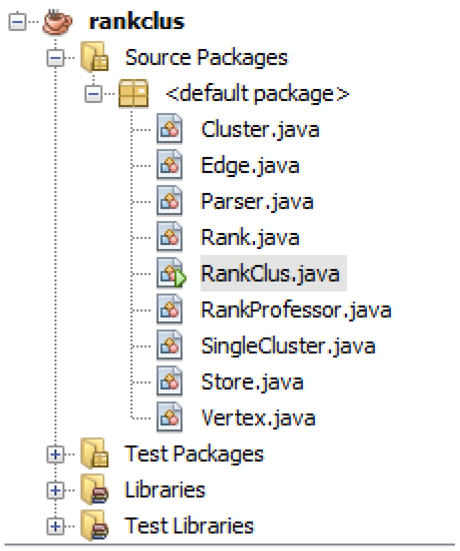


Figure 42: Classes of rankclus project

All that mentioned, the network schema generated abides with the bipartite network schema discussed in Chapter 4 (RankClus was mainly designed to work with bipartite networks).

According to the literature references, the whole implementation structure in this experiment could be easily expanded or modified to allow the utilization of other real world datasets that follow this specific network structure.

## Results of RankClus

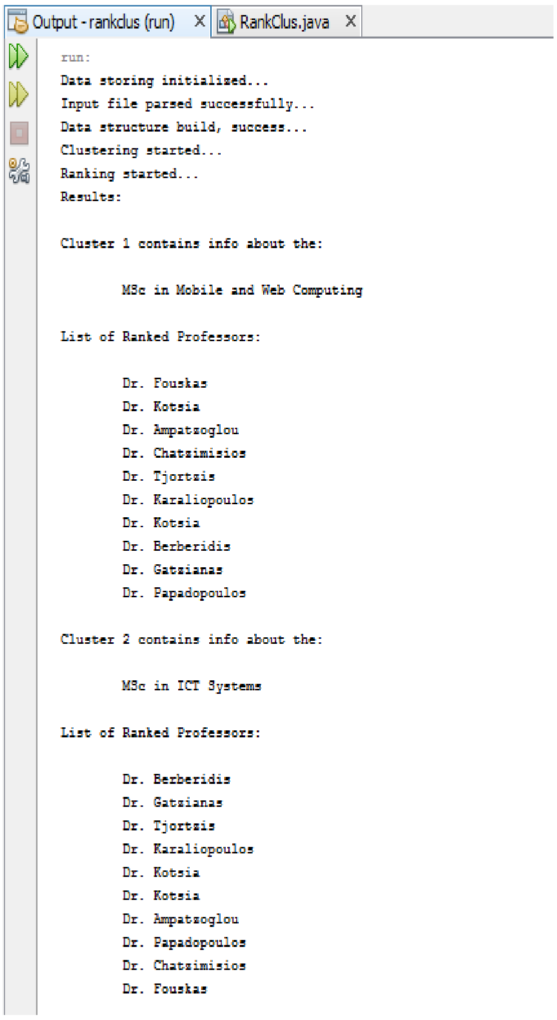
After compiling and running the java code produced in NetBeans it is considered necessary to present the results to evaluate the produced product's performance, correctness and precision in results.

Although correctness cannot be evaluated comparing with real world results, since the links professors-course and course-MSc programs were randomly generated.

As discussed in Chapter 3, ranking functions are very important since they can measure an entity's importance within a social network.

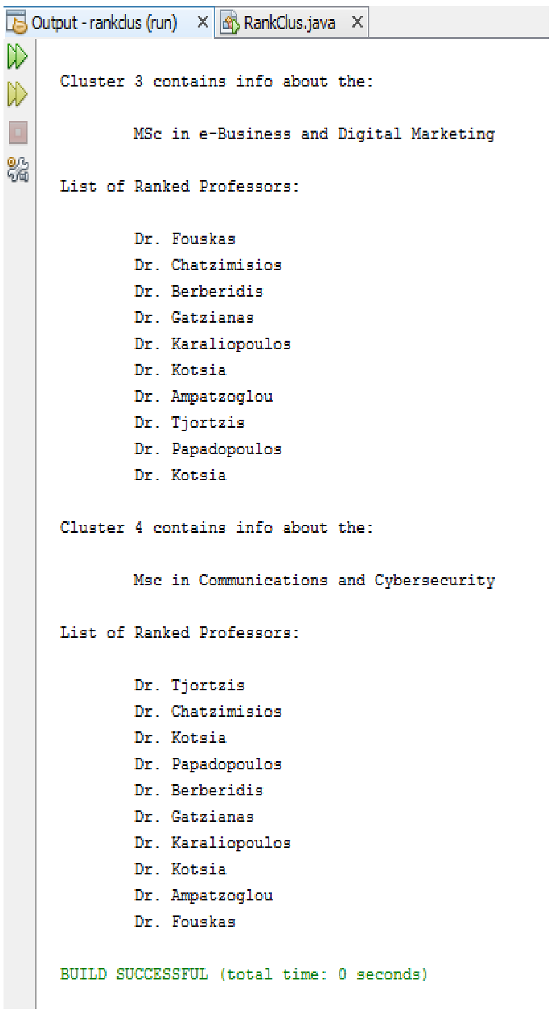
RankClus can create groups (clusters) of entities, but at the same time it can maintain the equality of importance both on ranking and clustering notions [37].

The following figures depict the output after running the rankclus project on NetBeans. It creates a cluster for each MSc program contained in the dataset provided and it calculates the ranking for each of the professors, repeating the iteration steps discussed earlier.



```
run:
Data storing initialized...
Input file parsed successfully...
Data structure build, success...
Clustering started...
Ranking started...
Results:

Cluster 1 contains info about the:

        MSc in Mobile and Web Computing

List of Ranked Professors:

        Dr. Fouskas
        Dr. Kotsia
        Dr. Ampatzoglou
        Dr. Chatzimisios
        Dr. Tjortzis
        Dr. Karaliopoulos
        Dr. Kotsia
        Dr. Berberidis
        Dr. Gatzianas
        Dr. Papadopoulos

Cluster 2 contains info about the:

        MSc in ICT Systems

List of Ranked Professors:

        Dr. Berberidis
        Dr. Gatzianas
        Dr. Tjortzis
        Dr. Karaliopoulos
        Dr. Kotsia
        Dr. Kotsia
        Dr. Ampatzoglou
        Dr. Papadopoulos
        Dr. Chatzimisios
        Dr. Fouskas
```

Figure 43: Output RankClus Clusters 1, 2



```
Cluster 3 contains info about the:

        MSc in e-Business and Digital Marketing

List of Ranked Professors:

        Dr. Fouskas
        Dr. Chatzimisios
        Dr. Berberidis
        Dr. Gatzianas
        Dr. Karaliopoulos
        Dr. Kotsia
        Dr. Ampatzoglou
        Dr. Tjortzis
        Dr. Papadopoulos
        Dr. Kotsia

Cluster 4 contains info about the:

        Msc in Communications and Cybersecurity

List of Ranked Professors:

        Dr. Tjortzis
        Dr. Chatzimisios
        Dr. Kotsia
        Dr. Papadopoulos
        Dr. Berberidis
        Dr. Gatzianas
        Dr. Karaliopoulos
        Dr. Kotsia
        Dr. Ampatzoglou
        Dr. Fouskas

BUILD SUCCESSFUL (total time: 0 seconds)
```

Figure 44: Output RankClus Clusters 3, 4

## 5.5 Contribution

This chapter discusses about a new methodology related to the following relatively uncharted field of studies: “local, user-guided clustering and ranking in Heterogeneous Information Networks”.

### 5.5.1 Introduction

We propose the UGRC (User-guided ranking-clustering) algorithm for local, user-guided clustering and ranking in HINs. Current approaches tackle the problem of global ranking and clustering. However, quite often, especially in very large graphs, users are interested in discovering knowledge in the close vicinity (neighborhood) of a specific node. The work done in this dissertation deals with the methodology followed for modeling HINs using a NoSQL DBMS in a way that will allow us to implement various mining tasks. UGRC belongs to the field of semi-supervised learning [33] where the results output is user specified.

### 5.5.2 Detailed Steps of UGRC algorithm

Our approach is theoretical, and it is based on the methodology and tools used on the previous experiments.

The algorithm functions in five (5) steps as described below:

*Step 1:* Database creation and population. We create a database and populate it using the methodology we followed in the previous experiments (e.g. we could use the Yelp database).

*Step 2:* User defines a query node *q* (e.g. using the Yelp database, we could request a restaurant with a specific name).

*Step 3:* Find a subgraph around node *n*, according to given constraints for its size. We apply the *k-nearest neighbor* algorithm requesting all the neighboring nodes from node *q*, thus, generating a subgraph of *n* neighbors, where *n* is a limiter for the number of neighboring nodes.

*Step 4:* Rank/cluster this subgraph. We use the RankClus algorithm [37] (as implemented in *Experiment 3)* to rank and cluster this subgraph generated from *Step 3*.

*Step 5:* Visualize the results. We could use custom generated functions within OrientDB (or Java code) to show the clusters and ranked lists produced after *Step 4*.

### 5.5.3 Discussion

We concluded to the proposed algorithm after referring to the literature trying to discover “grey” areas or novel methodologies that could effectively deal with the idea of user-guided local rank-clustering on HINs. The implementation and testing of this algorithm comprises one of the possible *short term* future works of this dissertation as summarized in chapter 7.2.2).

# 6 Tools

This chapter discusses the tools and programming languages used for experiments in terms of justification why choosing these specific ones. Links are supplied for a more thorough description of the available features of the tools [64] [60] [72].

## 6.1 Discussion about OrientDB

The main issues developers face when in process of implementing a software that utilizes a DBMS is the speed and flexibility it offers in terms of data accessing (e. g. reads/writes).

OrientDB is designed based on principles of supporting multiple models, being open source and a NoSQL DBMS. It combines the utilization of graphs and document flexibility into one single framework which is considered an operational database.

The main reasons for using OrientDB when handling Big Data for implementing the experiments of this dissertation are noted below:

I. *It is fast,*

OrientDB can import documents as all other DBMSs can, but can also utilize relations between objects. This is achieved using pointers (or indexes) that are appointed by default and are persistent for implementing very fast queries. Thus, no JOINs are required and data can be stored in the form of graphs where commands like SELECT or TRAVERSE can return a huge bulk of results near-instantly.

Also, with the era of Big Data at large, OrientDB handles relationships between objects as links (edges) having a unique identifier. In relational DBMSs when you input more data you become slower. OrientDB solves this issue as described above and makes it suitable for handling datasets of millions of records.

II. *It is highly scalable,*

Elastic linear scalability takes place, which is very important when you want to expand. With the commonly used master-slave architecture usually servers cannot handle the increasing requests. In OrientDB, throughput can scale up very easy just by adding an auto-configurated server to the working network. Also, auto-synch takes place allowing the newly-added server to be part of the server cluster. Downtimes do not exist, as well as time-wasting server configurations.

III. *It is open source,*

We state this fact referring to the population of the developers occupied by an open source project code; being accessible to everyone with no limitation and bug reporting, which are constants that can make a project "shine".

IV. *Queries using SQL query language,*

Last reason for rendering OrientDB the most suitable candidate is that the supported quivering language is based on SQL (although modified to handle graph and tree structures). SQL is the most commonly used querying language and there is huge support and usable paradigms and resources.

## 6.2 Discussion about Gephi

The main purpose of Gephi is to assist developers regarding data analytics that use graph structures. It offers a native infrastructure that allows to import data structures and form graph representations allowing to manipulate data and present them.

Also, with the use of this tool you can discover patterns in Big data after enforcing some hypothesis upon the data resources. It acts as a complementary tool that can work with our DBMS to assist us on calculating graph measures.

This tool is a project belonging to the Visual Analytics field of studies that aims to perform exploratory data analysis [75].

The main reasons for using Gephi for handling Big Data and implementing the experiments of this dissertation are noted below:

I. *It supports graph measures,*

It is the most important reason for using this tool, it will allow computing of graph metrics by querying our databases. It implements two (2) frameworks, the statistics and metrics framework which utilize the most common measures for social network analysis [92] as we discussed in this dissertation. Such measures are:

- Betweenness Centrality, Closeness, Diameter, Clustering Coefficient and PageRank,
- Community detection (Modularity),
- Random generators,
- Shortest path.

II. *It works with plugins,*

Gephi supports a plugin center that automatically updates and downloads plugins that are associated with our task at hand. Plugins are used by Gephi to extend its basic functionalities. For our dissertation, we want to utilize the metrics and statistics framework of Gephi so we use a two (2) step integration.

Our datasets are facilitated in localhost in OrientDB server. So, we use:

- a streaming plugin that calls the server through an HTTP request and then a new "/gephi" command in HTTP GET method is exposed which executes and returns a query in form of a result set in "gephi" format. Such queries where implemented in Chapter 5 (5.3.3 Implementation).
- The second step of integration is the Gephi importer for blueprints [16].

III. *It is open source,*

Same reason as discussed with OrientDB; we refer to the population of developers occupied by an open source project. Code being accessible to everyone with no limitation and bug reporting are constants that can make a project succeed. Also, the open source concept helps research evolve faster. This is achieved by having freely accessible resources aiding in enforcing faster improvements.

## 6.3 Discussion about Node.js

Node.js was used in one of the experiments as an alternative mean for achieving the same goal; the modeling, implementation and then, experimentation on a specific network schema. Combined with a browser, a database that supports json data and JSON can provide a unified development stack. For our purposes, Node.js was used as a parser to populate our database in OrientDB studio [64].

In detail, Node.js was used for implementing the experiments of this dissertation based on the next characteristics:

I. *The nature of its initial design,*

Node.js is used for development of network programs like Web servers. Its usability is like PHP [93], but their difference can be considered fundamental. Most often in PHP, executed functions must finish execution before a new command can be initiated, while in Node.js executions are designed to be executed in parallel with other commands and report back the status of finale (completion/failure).

In other words, PHP uses an architecture that each function blocks the execution of another, while Node.js comes to allow a parallel execution.

II. *Used as a parser of .js files,*

OrientDB can handle many types of dataset files (.csv, .json, etc) and the yelp dataset was supplied in .json format. So, using the single thread operational concept of Node.js and the non-blocking I/O calls, it seemed appropriate to implement a parser for populating (creating appropriate nodes and edges) the Yelp database in OrientDB.

III. *It is open source,*

Same as with OrientDB and Gephi there is a very large population of developers occupied by an open source project. Code being accessible to everyone with no limitation and bug reporting are constants that can make a project be a success.
Node.js uses the npm as its pre-installed package manager server platform, where thousands of developers can use to download and upload their work. Also, the open source concept helps research evolve faster. This is achieved by having freely accessible resources, aiding in enforcing faster improvements.

# 7 Results

This chapter makes points regarding the modeling, implementation, difficulties that arose during the whole procedure and discusses future work.

## 7.1 General Discussion

The present study investigates popular methodologies to deal with mining tasks on heterogeneous information networks. The work done tried to diagnose and cover the key points that can create the basic infrastructure for further researching on HINs. We focused on modeling the data following specific network schemas using datasets that are widely used by the academic/research community.

The study shows that such a modeling task can be achieved for complex information networks by using open source tools, which, if combined properly, it can be very effective.

These findings concur with other studies showing that mining HINs can be challenging. This claim is validated by the numerous novel implementations dedicated to this field of studies, during the recent years.

Knowledge discovery was achieved through querying our dataset and getting results without taking into consideration trivia information contained in the raw information or by deliberately excluding it, as well as by giving an example of implementing a hybrid algorithm (RankClus).

Most of the time was spent coding in Java and Node.js as well as utilizing the population of the databases generated for the experiments. We focused on specific points of the HIN topic such as implementation of the appropriate network schema (or transformation of datasets to abide with specific schemas) and the calculation of graph measures.

Although this study was conducted using stored-static data, the implementation should be generalized enabling the coverage of numerous complex information networks such as social media live data feed.

## 7.2 Future work

This section is split into two parts. During the process of writing this dissertation, extensive literature was studied but also experiments with tools were conducted. Thus, possible new frontiers are presented that need to be advertised by the research community about HINs and social media mining, as well as possible improvements and future work associated with the experiments.

### 7.2.1 Future work deriving from literature

Having considered numerous HIN books, research papers and literature about complex information networks and social mining, a list of probable relatively unchartered research frontiers is presented [49]. These research topics can be modeled and experimented on, using as a foundation to build on, the methodology followed during this dissertation.

I. Modeling large information networks as HINs and discovering knowledge from the model structure. There are numerous algorithms or methodologies being developed regarding HINs most of them described in the literature section. Most common concepts are clustering, classification, ranking, pattern discovery where the ultimate aim is to explore the power of links (edges) and conclude to logical facts from that redundant information.

II. Another emerging topic regarding HINs is similarity search and OLAP. These concepts are not new but applying them to information networks such as HINs requires heavy revamps. Other topics consistent with this research frontier are summarization of information networks, indexing and cube materialization of such complex networks.

III. Handling social media data and creating temporal information networks might be tricky. Research could be conducted aiming on handling temporal information of large-scale networks. Finding trends, outliers and patterns based on timestamps is a research frontier with numerous applications.

IV. With the IOTs (Internet of Things) at large and all the data produced and stored into sensor networks, another connection could be established linking sensor networks with information networks’ entities. The product could be a cyber-physical network that could implement any kind of knowledge discovery concept discussed in this dissertation.

V. HINs become more and more common nowadays. Nodes and links in such networks can also contain text. In social media sites, we find blogs, descriptions, discussions, documents etc. which can be linked with other similar-typed objects (or not) forming text-based information networks. So, a new era begins that thirsts for effective text search, information analysis (in text) that is utilized in HINs. Research focused on this topic could be encouraged.

VI. Due to the era of social media mining, we are in dire need to handle information networks that are too rich in terms of the diversity of objects they include, as well as connections established between them. We most often handle semi-structured data; thus, great endeavor could be made to enhance the information network analysis methodologies so that they can handle large databases having the specifications described more effectively.

VII. During reviewing the available literature concerning HINs, it was discovered that minor research has been conducted that connects HINs with web mining. Methods could be implemented covering this topic, merging the information network analysis with web data which is perceived as multiple interconnected networks of k-typed objects. Enforcing analytics in those data could enable us to discover knowledge in vast databases.

VIII. Once more, in the field of information network analysis, the methods used could be improved so that the often-interconnected information found in social networks could be handled more intelligently. Such procedures are data cleaning and data validation which is observed to suffer when dealing with social network data.

### 7.2.2 Future work arising from experiments

After implementing our task and having discussed our results, another future work section arises that can be directly linked with our findings. It is more like a list of add-ons to the existing implementation (Experiments 1, 2, 3). One could use the code from the experiments of this dissertation and enrich it aiming on covering the following topics:

I. Implement embedded functions utilizing graph metrics in OrientDB studio. By doing that the use of Gephi could be eliminated. The OrientDB supports the following function features:

- Once created they persist, meaning that they are saved and stick with the database.

- They can be coded in Javascript (in the future in Ruby, Scala and Java programming languages according to OrientDB documentation [16]).
- They support plugins which can inject new objects for use by functions.
- Each function can call other functions.

II. Modify/Improve RankClus code written in Java to handle other types of network schemas (currently it is mostly used for bipartite networks) widely expanding its functionality.

III. Create a Java package implementing novel hybrid algorithms (like the ones discussed in this dissertation) that can be used together with the OrientDB Graph API. This would act as an add-on on OrientDB that would allow users to manage social media networks using a single powerful tool.

IV. Implement (or improve) an intelligent Parser (e.g enhancing ETL module) to handle large raw data imports of any type, appropriate for social media information networks. This idea would also act as an add-on on OrientDB aiming on the utilization of a single powerful tool.

V. Implement an algorithm for local, user-guided clustering and ranking in HINs. Current research deals with the issue of global ranking and clustering. But sometimes, especially in huge graphs, users want to discover knowledge concerning the neighborhood of a user-defined node.

# 8 Conclusion

During the last years, a growing tendency on understanding heterogeneous information networks has appeared. This can only be interpreted as an outcome of the rich structural and semantic information that exists in this kind of networks. To achieve in withdrawing knowledge from these vast sources of data, we need to improve existing ones or develop novel methodologies for handling appropriate mining tasks.

This dissertation aims to report some of the findings as well as state of the art methodologies of this rapidly expanding field of data mining. Literature is excessively addressed and advised, presenting facts to support reasoning during the implementation part of this work.

Latest findings and developments of heterogeneous information networks along with lab testing of methodologies and algorithms of graph mining is conducted. We aim to show basic issues that we might face while dealing with knowledge discovery in heterogeneous information networks.

We focus on specific aspects of that issues, such as appropriate network schemas depending on the mining task, as well as tools for effective implementation of modeling and graph measures' calculation.

This is achieved with the generation of three (3) experiments and the contribution chapter, that each one has a different purpose which is summarized as:

*Experiment 1:* Utilization of a bipartite schema. Modeling and population of a database abiding with that schema and testing the validity of the model using queries.

*Experiment 2:* Utilization of a star-schema network. Modeling, population of a database, implementation of queries and calculation of graph measures.

*Experiment 3:* Generation of a synthetic graph that can work with a specific algorithm, utilizing ranking and clustering together and presentation of results.

*Contribution:* Proposition of a new methodology for user-guided local ranking and clustering on Heterogeneous Information Networks.

Finally, the implementation part is presented in a detailed way that would allow any reader to understand and properly utilize the methodologies and source code accompanying this dissertation. Also, thorough presentation of future work was made, both research and implementation oriented.

# Bibliography

[1] K. Selc¸uk Candan and Maria Luisa. Sapino, Data management for multimedia retrieval., Cambridge University Press, 2010.

[2] Andreas M. Kaplan and Michael. Haenlein, Users of the world, unite! the challenges and opportunities of social media., Business horizons **53** (2010), no. 1, 59–68.

[3] Pritam Gundecha and Huan. Liu, Mining social media: a brief introduction., Tutorials in Operations Research **1** (2012).

[4] Nicole B. Ellison and et al., Social network sites: definition, history, and scholarship., Journal of Computer-Mediated Communication **13** (2007).

[5] Reza Zafarani, Mohammad Ali Abbasi, Huan Liu, Social Media Mining: An introduction, Cambridge University Press, Draft Version (2014).

[6] Chistopher M. Bishop, Pattern recognition and machine learning, Springer, 2006.

[7] Richard O. Duda, Peter E. Hart, and David G. Stork, Pattern classification., Wiley-interscience, 2012.

[8] D. Easley and J.M. Kleinberg, Networks, crowds, and markets., Cambridge University Press, 2010.

[9] John. Scott, Social network analysis., Sociology **22** (1988), no. 1, 109–127.

[10] Lei Tang, XufeiWang, and Huan. Liu, Community detection via heterogeneous interaction analysis., Data Mining and Knowledge Discovery (DMKD) **25** (2012), no. 1, 1– 33.

[11] Jan H. Kietzmann, Kristopher Hermkens, Ian P. McCarthy, and Bruno S. Silvestre, Social media? get serious! understanding the functional building blocks of social media., BusinessHorizons **54** (2011), no. 3, 241–251.

[12] Lorrain, F. & White, H.C. 1971. Structural equivalence of individuals in social networks. Journal of Mathematical Sociology 1: 49-80.

[13] Li, Guoliang, et al. "EASE: an effective 3-in-1 keyword search method for unstructured, semi-structured and structured data." *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*. ACM, 2008.

[14] Digital image. *Altova*. Web. 2, (2016, November 2). Retrieved from: https://www.altova.com/mapforce/data-sorting.html.

[15] Reza Zafarani, Mohammad Ali Abbasi, Huan Liu, Social Media Mining: An introduction, Cambridge University Press, Draft Version (2014), 16-17.

[16] Documentation of orientDB, (2016, November 10) Retrieved from: http://orientdb.com/docs/.

[17] Statistics for Facebook, Twitter, Youtube, Flickr, Wikipedia. (2016, October 31). Retrieved from https://www.statista.com/search/.

[18] Moosavi, Seyed Ahmad, et al. "Community detection in social networks using user frequent pattern mining." *Knowledge and Information Systems* (2016): 1-28.

[19] Sinkovits, Robert S., et al. "Fast determination of structurally cohesive subgroups in large networks." Journal of Computational Science (2016).

[20] Plantié, Michel, and Michel Crampes. "Survey on social community detection."*Social media retrieval*. Springer London, 2013. 65-85.

[21] Papadopoulos, Symeon, et al. "Community detection in social media." *Data Mining and Knowledge Discovery* 24.3 (2012): 515-554.

[22] Chang, Maw-Shang, et al. "Finding large k-clubs in undirected graphs."*Computing* 95.9 (2013): 739-758.

[23] P. Jaccard, Distribution de la Flore Alpine: dans le Bassin des dranses et dans quelques r´egions voisines., Rouge, 1901.

[24] Rawashdeh, A.; Rawashdeh, M.; D´ıaz, I.; and Ralescu, A. 2014. Measures of semantic similarity of nodes in a social network. In Information Processing and Management of Uncertainty in Knowledge-Based Systems, 76–85. Springer.

[25] Amini, Arash A., et al. "Pseudo-likelihood methods for community detection in large sparse networks." *The Annals of Statistics* 41.4 (2013): 2097-2122.

[26] Ardilly, Pascal, et al. "Les Techniques de Sondage." *Technometrics* 57.2 (2015).

[27] Newman, M. E. J. "Community detection in networks: Modularity optimization and maximum likelihood are equivalent." *arXiv preprint arXiv:1606.02319* (2016).

[28] Tsourakakis, Charalampos, et al. "Denser than the densest subgraph: extracting optimal quasi-cliques with quality guarantees." *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2013.

[29] Anderson, Rodney. "The Insight to the Girvan-Newman algorithm." (2016).

[30] Thomas, Juan Carlos Rojas. "New version of Davies-Bouldin index for clustering validation based on cylindrical distance." *V Chilean Workshop on Pattern Recognition*. 2013.

[31] Navigli, Roberto, and Daniele Vannella. "Semeval-2013 task 11: Word sense induction and disambiguation within an end-user application." *Second Joint Conference on Lexical and Computational Semantics (* SEM)*. Vol. 2. 2013.

[32] Jenneth, A., and K. Thangavel. "Dimensionality Reduction based on Hubness Property using Feature Weighting Method for Clustering." *Indian Journal of Science and Technology* 9.19 (2016).

[33] Chuan Shi, Yitong Li, Jiawei Zhang, Yizhou Sun, Philip S. Yu, A Survey of Heterogeneous Information Network Analysis, IEEE Transcactions on knowledge and data Engineering (2015).

[34] Mikko Kivelä, Alexandre Arenas, Marc Barthelemy, James P. Gleeson, Yamir Moreno, Mason A. Porter*, Multilayer Networks, 2014.

[35] J. Han, "Mining heterogeneous information networks by exploring the power of links," in Discovery Science, 2009, pp. 13–30.

[36] Y. Sun and J. Han, "Mining heterogeneous information networks: a structural analysis approach," SIGKDD Explorations, vol. 14, no. 2, pp. 20–28, 2012.

[37] Y. Sun, J. Han, P. Zhao, Z. Yin, H. Cheng, and T. Wu, "RankClus: integrating clustering with ranking for heterogeneous information network analysis," in EDBT, 2009, pp. 565–576.

[38] Y. Sun, J. Han, X. Yan, P. Yu, and T. Wu, "Pathsim: Meta path-based top-k similarity search in heterogeneous information networks," in VLDB, 2011, pp. 992–1003.

[39] C. Shi, X. Kong, P. S. Yu, S. Xie, and B. Wu, "Relevance search in heterogeneous networks," in EDBT, 2012, pp. 180–191.

[40] Y. Sun, B. Norick, J. Han, X. Yan, P. S. Yu, and X. Yu, "Integrating meta-path selection with user-guided object clustering in heterogeneous information networks," in KDD, 2012, pp. 1348–1356.

[41] X. Kong, P. S. Yu, Y. Ding, and D. J. Wild, "Meta path-based collective classification in heterogeneous information networks," in CIKM, 2012, pp. 1567–1571.

[42] Yizhou Sun and Jiawei Han, Mining Heterogeneous Information Networks: Principles and Methodologies, Synthesis lectures on data mining and knowledge discovery #5, Morgan & Claypool, 2012.

[43] Reza Zafarani, Mohammad Ali Abbasi, Huan Liu, Social Media Mining: An introduction, Cambridge University Press, Draft Version (2014), 105-129.

[44] R.W. Floyd, Algorithm 97: shortest path, Communications of the ACM **5** (1962), no. 6, 345.

[45] Computer science bibliography (dblp), (2016, October 15) Retrieved from: http://dblp.uni-trier.de/.

[46] Watts, Duncan J.; Strogatz, Steven H. (June 1998). "Collective dynamics of 'small-world' networks". *Nature*. **393** (6684): 440–442.

[47] D.J. Watts, Networks, dynamics, and the small-world phenomenon., American Journal of Sociology **105** (1999), no. 2, 493–527.

[48] A.L. Barabasi and R. Albert, Emergence of scaling in random networks., science **286** (1999), no. 5439, 509–512.

[49] Information network analysis and data mining promising research directions. (2016, November 15). Retrieved from http://hanj.cs.illinois.edu/projs/infonet.htm.

[50] R.Y. Wang, V.C. Storey, C.P. Firth, *A Framework for Analysis of Data Quality Research,* IEEE Transactions on Knowledge and Data Engineering 7 (1995) 623-640 doi: 10.1109/69.404034.

[51] X. Zhu, X. Wu, *Class Noise vs. Attribute Noise: A Quantitative Study,* Artificial Intelligence Review 22 (2004) 177-210 doi: 10.1007/s10462-004-0751-8.

[52] Hui Xiong, Gaurav Pandey, Michael Steinbach, Vipin Kumar, Enhancing Data Analysis with Noise Removal, IEEE Transactions on Knowledge and Data Engineering (TKDE), Vol. 18, No. 3, pp. 304-319, March 2006.

[53] Petrovskiy, Outlier Detection Algorithms in Data Mining Systems, M.I. Programming and Computer Software (2003) 29: 228. doi:10.1023/A:1024974810270.

[54] Singh Vijendra and Pathak Shivani, Robust Outlier Detection Technique in Data Mining: A Univariate Approach, Faculty of Engineering and Technology Mody Institute of Technology and Science Lakshmangarh, Sikar, Rajasthan, India, 2014.

[55] Jerzy W. Grzymala-Busse, Witold J. Grzymala-Busse, Handling Missing Attribute Values, Data Mining and Knowledge Discovery Handbook Chapter 3, pp 33-51, July 2010.

[56] J. Jebamalar Tamilselvi and C. Brilly Gifta, Handling Duplicate Data in Data Warehouse for Data Mining, International Journal of Computer Applications (0975 – 8887) Volume 15– No.4, February 2011.

[57] Bodnar, Todd, and Marcel Salathé. "Validating models for disease detection using twitter." *Proceedings of the 22nd International Conference on World Wide Web*. ACM, 2013.

[58] Witten, Ian H. "Data Mining with WEKA." (2013).

[59] J. Lafferty, A. McCallum, and F. C. Pereira, "Conditional random fields: Probabilistic models for segmenting and labeling sequence data," in ICML, 2001, pp. 282–289.

[60] Framework analytical description, (2016, October 25) Retrieved from: http://orientdb.com/orientdb/.

[61] X. Kong, B. Cao, and P. S. Yu, "Multi-label classification by mining label and instance correlations from heterogeneous information networks," in KDD, 2013, pp. 614–622.

[62] Y. Zhou and L. Liu, "Activity-edge centric multi-label classification for mining heterogeneous information networks," in KDD, 2014, pp. 1276–1285.

[63] S. Jendoubi, A. Martin, L. Lietard, and B. B. Yaghlane, "Classification of message spreading in a heterogeneous social network," in IPMU, 2014, pp. 66–75.

[64] Product analytical description, (2016, November 5) Retrieved from: https://nodejs.org/en/.

[65] Choi, Seung-Seok, Sung-Hyuk Cha, and Charles C. Tappert. "A survey of binary similarity and distance measures." *Journal of Systemics, Cybernetics and Informatics* 8.1 (2010): 43-48.

[66] A. K. Jain, "Data clustering: 50 years beyond k-means," Pattern Recognition Letters, vol. 31, no. 8, pp. 651–666, 2010.

[67] Reza Zafarani, Mohammad Ali Abbasi, Huan Liu, Social Media Mining: An introduction, Cambridge University Press, Draft Version (2014), pp. 162-165.

[68] J. Shi and J. Malik, "Normalized cuts and image segmentation," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 22, no. 8, pp. 888–905, 2000.

[69] Y. Zhou, H. Cheng, and J. X. Yu, "Graph clustering based on structural/attribute similarities," in VLDB, 2009, pp. 718–729.

[70] M. Sales-Pardo, R. Guimera, A. Moreira, and L. Amaral, "Extracting the hierarchical organization of complex systems," Proceedings of the National Academy of Sciences, vol. 104, no. 39, pp. 15 224–15 229, 2007.

[71] Github open source code projects, (2016, September 5) Retrieved from: https://github.com/.

[72] Gephi Product features, (2016, October 7) Retrieved from: https://gephi.org/.

[73] D. Liben-Nowell and J. Kleinberg, “The link-prediction problem for social networks,” Journal of the American society for information science and technology, vol. 58, no. 7, pp. 1019–1031, 2007.

[74] A. Popescul and L. H. Ungar, “Statistical relational learning for link prediction,” in IJCAI workshop on learning statistical models from relational data, 2003.

[75] Keim, Daniel, et al. "Visual analytics: Definition, process, and challenges." Information visualization. Springer Berlin Heidelberg, 2008.

[76] Y. Sun, Y. Yu, and J. Han, “Ranking-based clustering of heterogeneous information networks with star network schema,” in KDD, 2009, pp. 797–806.

[77] Chuan Shi, Yitong Li, Jiawei Zhang, Yizhou Sun, Philip S. Yu, A Survey of Heterogeneous Information Network Analysis, IEEE Transcactions on knowledge and data Engineering (2015), 15-17.

[78] C. Luo, W. Pang, Z. Wang, and C. Lin, “Hete-cf: Social-based collaborative filtering recommendation using heterogeneous relations,” in ICDM, 2014.

[79] N. Srebro, T. Jaakkola et al., “Weighted low-rank approximations,” in ICML, 2003, pp. 720–727.

[80] X. Yang, H. Steck, and Y. Liu, “Circle-based recommendation in online social networks,” in KDD, 2012, pp. 1267–1275.

[81] Y.-K. Shih and S. Parthasarathy, “Scalable global alignment for multiple biological networks,” BMC bioinformatics, vol. 13, no. Suppl 3, p. S11, 2012.

[82] A. Doan, J. Madhavan, P. Domingos, and A. Halevy, “Ontology matching: A machine learning approach,” in Handbook on ontologies, 2004, pp. 385–403.

[83] L. Page, S. Brin, R. Motwani, and T. Winograd, the pagerank citation ranking: bringing order to the web, (1999).

[84] Sabidussi, G.: The centrality index of a graph. Psychometrika 31(4), 581-603 (1966).

[85] Freeman, L.: A set of measures of centrality based on betweenness. Sociometry 40(1), 35-41 (1977).

[86] L. C. Freeman, “Centrality in social networks conceptual clarification,” Social Networks, vol. 1, no. 3, pp. 215–239, Jan. 1978.

[87] P. D. Straffin, Jr, Algebra in Geography: Eigenvectors of Networks, Mathematics Magazine, Vol.53, No. 5(Nov., 1980), 269-276.

[88] Chaoqun Ni, Cassidy R. Sugimoto, and Jiepu Jiang Jan 2011, Degree, Closeness and Betweenness: Application of group centrality measurements to explore macro disciplinary evolution diachronically, ResearchGate: https://www.researchgate.net/publication/228942246.

[89] Leo Katz: A New Status Index Derived from Sociometric Index. Psychometrika 18(1):39–43, 1953http://phya.snu.ac.kr/~dkim/PRL87278701.pdf.

[90] Reza Zafarani, Mohammad Ali Abbasi, Huan Liu, Social Media Mining: An introduction, Cambridge University Press, Draft Version (2014), 87-92.

[91] Reza Zafarani, Mohammad Ali Abbasi, Huan Liu, Social Media Mining: An introduction, Cambridge University Press, Draft Version (2014), 92-95.

[92] Scott, John. *Social network analysis*. Sage, 2012.

[93] Arora, Ms Pooja, Anurag Dixit, and Delhi BCIIT. "Analysis of Cloud IDEs for Software Development." In International Journal of Engineering Research and General Science Volume 4, Issue 4, July-August, 2016, ISSN 2091-2730.

[94] Ulrik Brandes, A Faster Algorithm for Betweenness Centrality, in Journal of Mathematical Sociology 25(2):163-177, (2001).

[95] Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, Etienne Lefebvre, Fast unfolding of communities in large networks, in Journal of Statistical Mechanics: Theory and Experiment 2008 (10), P1000.

# Appendix

## Yelp Dataset

Each file is composed of a single object type, one json-object per-line. There are three (3) .json files. For users, reviews and businesses describing each object as shown below.

*yelp_academic_dataset_user.json*

```
{
   'type': 'user',
   'user_id': (encrypted user id),
   'name': (first name),
   'review_count': (review count),
   'average_stars': (floating point average, like 4.31),
   'votes': {(vote type): (count)},
   'friends': [(friend user_ids)],
   'elite': [(years_elite)],
   'yelping_since': (date, formatted like '2012-03'),
   'compliments': {
      (compliment_type): (num_compliments_of_this_type),
      ...
   },
   'fans': (num_fans),
}
```

*yelp_academic_dataset_review.json*

```
{
   'type': 'review',
   'business_id': (encrypted business id),
   'user_id': (encrypted user id),
   'stars': (star rating, rounded to half-stars),
   'text': (review text),
   'date': (date, formatted like '2012-03-14'),
   'votes': {(vote type): (count)},
}
```

*yelp_academic_dataset_business.json*

```
{
    'type': 'business',
    'business_id': (encrypted business id),
    'name': (business name),
    'neighborhoods': [(hood names)],
    'full_address': (localized address),
    'city': (city),
    'state': (state),
    'latitude': latitude,
    'longitude': longitude,
    'stars': (star rating, rounded to half-stars),
    'review_count': review count,
    'categories': [(localized category names)]
    'open': True / False (corresponds to closed, not business hours),
    'hours': {
        (day_of_week): {
            'open': (HH:MM),
            'close': (HH:MM)
        },
        ...
    },
    'attributes': {
        (attribute_name): (attribute_value),
        ...
    },
}
```

## Movies Dataset Description

This dataset (ml-latest) describes a 5-star rating and free-text tagging activity from MovieLens (http://movielens.org), a movie recommendation service. It contains 24404096 ratings and 668953 tag applications across 40110 movies. These data were created by 259137 users between January 09, 1995 and October 17, 2016. This dataset was generated on October 18, 2016. Users were selected at random for inclusion. All selected users had rated at least 1 movies. No demographic information is included. Each user is represented by an id, and no other information is provided.

*User Ids*

MovieLens users were selected at random for inclusion. Their ids have been anonymized. User ids are consistent between `ratings.csv` and `tags.csv` (i.e., the same id refers to the same user across the two files).

*Movie Ids*

Only movies with at least one rating or tag are included in the dataset. These movie ids are consistent with those used on the MovieLens web site (e.g., id `1` corresponds to the URL <https://movielens.org/movies/1>). Movie ids are consistent between `ratings.csv`, `tags.csv`, `movies.csv`, and `links.csv` (i.e., the same id refers to the same movie across these four data files).

*Ratings Data File Structure (ratings.csv)*

All ratings are contained in the file `ratings.csv`. Each line of this file after the header row represents one rating of one movie by one user, and has the following format:

userId, movieId, rating, timestamp

The lines within this file are ordered first by userId, then, within user, by movieId.

Ratings are made on a 5-star scale, with half-star increments (0.5 stars - 5.0 stars). Timestamps represent seconds since midnight Coordinated Universal Time (UTC) of January 1, 1970.

*Tags Data File Structure (tags.csv)*

All tags are contained in the file `tags.csv`. Each line of this file after the header row represents one tag applied to one movie by one user, and has the following format:

userId, movieId, tag, timestamp

The lines within this file are ordered first by userId, then, within user, by movieId.

Tags are user-generated metadata about movies. Each tag is typically a single word or short phrase. The meaning, value, and purpose of a particular tag is determined by each user. Timestamps represent seconds since midnight Coordinated Universal Time (UTC) of January 1, 1970.

*Movies Data File Structure (movies.csv)*

Movie information is contained in the file `movies.csv`. Each line of this file after the header row represents one movie, and has the following format:

movieId, title, genres

Movie titles are entered manually or imported from <https://www.themoviedb.org/>, and include the year of release in parentheses. Errors and inconsistencies may exist in these titles.

Genres are a pipe-separated list, and are selected from the following:

* Action, * Adventure, * Animation, * Children's, * Comedy, * Crime, * Documentary, * Drama, * Fantasy, * Film-Noir, * Horror, * Musical, * Mystery, * Romance, * Sci-Fi, * Thriller, * War, * Western, * (no genres listed)

*Links Data File Structure (links.csv)*

Identifiers that can be used to link to other sources of movie data are contained in the file `links.csv`. Each line of this file after the header row represents one movie, and has the following format:

movieId, imdbId, tmdbId

movieId is an identifier for movies used by <https://movielens.org>. E.g., the movie Toy Story has the link <https://movielens.org/movies/1>.

imdbId is an identifier for movies used by <http://www.imdb.com>. E.g., the movie Toy Story has the link <http://www.imdb.com/title/tt0114709/>.

tmdbId is an identifier for movies used by <https://www.themoviedb.org>. E.g., the movie Toy Story has the link <https://www.themoviedb.org/movie/862>.

*Citation: F. Maxwell Harper and Joseph A. Konstan. 2015. The MovieLens Datasets: History and Context. ACM Transactions on Interactive Intelligent Systems (TiiS) 5, 4, Article 19 (December 2015), 19 pages. DOI=<http://dx.doi.org/10.1145/2827872*

## PROFDB Dataset Description

This dataset represents a synthetic dataset that was created for testing an implementation of the rankclus (2009) algorithm. It contains professors, MSc programs, courses they teach and the links between courses with MSc programs as well as, professors with courses.

```
<?xml version="1.0" encoding="UTF-8"?>
<PROFDB>
<!-->initialize objects with ids </-->
  <OBJECTS>
    <OBJECT ID="11"/>
    <OBJECT ID="12"/>
    <OBJECT ID="13"/>
    <OBJECT ID="14"/>
    <OBJECT ID="15"/>
    <OBJECT ID="16"/>
    <OBJECT ID="17"/>
    <OBJECT ID="18"/>
    <OBJECT ID="19"/>
    <OBJECT ID="20"/>
    <OBJECT ID="21"/>
    <OBJECT ID="22"/>
    <OBJECT ID="23"/>
    <OBJECT ID="24"/>
    <OBJECT ID="25"/>
    <OBJECT ID="26"/>
    <OBJECT ID="27"/>
    <OBJECT ID="28"/>
    <OBJECT ID="29"/>
    <OBJECT ID="30"/>
    <OBJECT ID="31"/>
    <OBJECT ID="32"/>
    <OBJECT ID="33"/>
    <OBJECT ID="34"/>
    <OBJECT ID="35"/>
    <OBJECT ID="36"/>
    <OBJECT ID="37"/>
    <OBJECT ID="38"/>
    <OBJECT ID="39"/>
    <OBJECT ID="40"/>
    <OBJECT ID="41"/>
    <OBJECT ID="42"/>
    <OBJECT ID="43"/>
    <OBJECT ID="44"/>
    <OBJECT ID="45"/>
    <OBJECT ID="46"/>
  <OBJECT ID="47"/>
  <OBJECT ID="48"/>
  <OBJECT ID="49"/>
  <OBJECT ID="50"/>
  <OBJECT ID="51"/>
  <OBJECT ID="52"/>
  <OBJECT ID="53"/>
  <OBJECT ID="54"/>
  <OBJECT ID="55"/>
  <OBJECT ID="56"/>
  <OBJECT ID="57"/>
  <OBJECT ID="58"/>
  <OBJECT ID="59"/>
  <OBJECT ID="60"/>
</OBJECTS>

<LINKS>
  <!-->links courses-Msc programs</-->
  <LINK ID="1" O1-ID="21" O2-ID="41"/>
  <LINK ID="2" O1-ID="22" O2-ID="42"/>
  <LINK ID="3" O1-ID="23" O2-ID="43"/>
  <LINK ID="4" O1-ID="24" O2-ID="44"/>
  <LINK ID="5" O1-ID="25" O2-ID="45"/>
  <LINK ID="6" O1-ID="26" O2-ID="46"/>
  <LINK ID="7" O1-ID="27" O2-ID="47"/>
  <LINK ID="8" O1-ID="28" O2-ID="48"/>
  <LINK ID="9" O1-ID="29" O2-ID="49"/>
  <LINK ID="10" O1-ID="30" O2-ID="50"/>
  <LINK ID="11" O1-ID="11" O2-ID="51"/>
  <LINK ID="12" O1-ID="12" O2-ID="52"/>
  <LINK ID="13" O1-ID="13" O2-ID="53"/>
  <LINK ID="14" O1-ID="14" O2-ID="54"/>
  <LINK ID="15" O1-ID="15" O2-ID="55"/>
  <LINK ID="16" O1-ID="16" O2-ID="56"/>
  <LINK ID="17" O1-ID="17" O2-ID="57"/>
  <LINK ID="18" O1-ID="18" O2-ID="58"/>
  <LINK ID="19" O1-ID="19" O2-ID="59"/>
  <LINK ID="20" O1-ID="20" O2-ID="60"/>
     <!-->links : Professor-courses </-->
  <LINK ID="21" O1-ID="31" O2-ID="21"/>
  <LINK ID="22" O1-ID="32" O2-ID="22"/>
  <LINK ID="23" O1-ID="33" O2-ID="23"/>
  <LINK ID="24" O1-ID="34" O2-ID="24"/>
  <LINK ID="25" O1-ID="35" O2-ID="25"/>
  <LINK ID="26" O1-ID="36" O2-ID="26"/>
  <LINK ID="27" O1-ID="37" O2-ID="27"/>
  <LINK ID="28" O1-ID="38" O2-ID="28"/>
  <LINK ID="29" O1-ID="39" O2-ID="29"/>
  <LINK ID="30" O1-ID="40" O2-ID="30"/>
  <LINK ID="31" O1-ID="32" O2-ID="11"/>
  <LINK ID="32" O1-ID="33" O2-ID="12"/>
  <LINK ID="33" O1-ID="34" O2-ID="13"/>
  <LINK ID="34" O1-ID="35" O2-ID="14"/>
  <LINK ID="35" O1-ID="36" O2-ID="15"/>
  <LINK ID="36" O1-ID="37" O2-ID="16"/>
  <LINK ID="37" O1-ID="38" O2-ID="17"/>
  <LINK ID="38" O1-ID="39" O2-ID="18"/>
  <LINK ID="39" O1-ID="40" O2-ID="19"/>
  <LINK ID="40" O1-ID="31" O2-ID="20"/>
  <LINK ID="41" O1-ID="33" O2-ID="11"/>
  <LINK ID="42" O1-ID="34" O2-ID="12"/>
  <LINK ID="43" O1-ID="35" O2-ID="13"/>
  <LINK ID="44" O1-ID="36" O2-ID="14"/>
  <LINK ID="45" O1-ID="37" O2-ID="15"/>
  <LINK ID="46" O1-ID="38" O2-ID="16"/>
  <LINK ID="47" O1-ID="39" O2-ID="17"/>
  <LINK ID="48" O1-ID="40" O2-ID="18"/>
  <LINK ID="49" O1-ID="31" O2-ID="19"/>
  <LINK ID="50" O1-ID="32" O2-ID="20"/>
  <LINK ID="51" O1-ID="40" O2-ID="21"/>
  <LINK ID="52" O1-ID="31" O2-ID="22"/>
  <LINK ID="53" O1-ID="32" O2-ID="23"/>
  <LINK ID="54" O1-ID="33" O2-ID="24"/>
  <LINK ID="55" O1-ID="34" O2-ID="25"/>
  <LINK ID="56" O1-ID="35" O2-ID="26"/>
  <LINK ID="57" O1-ID="36" O2-ID="27"/>
  <LINK ID="58" O1-ID="37" O2-ID="28"/>
  <LINK ID="59" O1-ID="38" O2-ID="29"/>
  <LINK ID="60" O1-ID="39" O2-ID="30"/>
</LINKS>

<ATTRIBUTES>
  <!-->initialize all attributes </-->
  <ATTRIBUTE NAME="object-type">
    <ATTR-VALUE ITEM-ID="11">
      <COL-VALUE>course</COL-VALUE>
    </ATTR-VALUE>
    <ATTR-VALUE ITEM-ID="12">
      <COL-VALUE>course</COL-VALUE>
    </ATTR-VALUE>
    <ATTR-VALUE ITEM-ID="13">
      <COL-VALUE>course</COL-VALUE>
    </ATTR-VALUE>
    <ATTR-VALUE ITEM-ID="14">
      <COL-VALUE>course</COL-VALUE>
    </ATTR-VALUE>
```

```
<ATTR-VALUE ITEM-ID="15">
  <COL-VALUE>course</COL-VALUE>
</ATTR-VALUE>
<ATTR-VALUE ITEM-ID="16">
  <COL-VALUE>course</COL-VALUE>
</ATTR-VALUE>
<ATTR-VALUE ITEM-ID="17">
  <COL-VALUE>course</COL-VALUE>
</ATTR-VALUE>
<ATTR-VALUE ITEM-ID="18">
  <COL-VALUE>course</COL-VALUE>
</ATTR-VALUE>
<ATTR-VALUE ITEM-ID="19">
  <COL-VALUE>course</COL-VALUE>
</ATTR-VALUE>
<ATTR-VALUE ITEM-ID="20">
  <COL-VALUE>course</COL-VALUE>
</ATTR-VALUE>
<ATTR-VALUE ITEM-ID="21">
  <COL-VALUE>course</COL-VALUE>
</ATTR-VALUE>
<ATTR-VALUE ITEM-ID="22">
  <COL-VALUE>course</COL-VALUE>
</ATTR-VALUE>
<ATTR-VALUE ITEM-ID="23">
  <COL-VALUE>course</COL-VALUE>
</ATTR-VALUE>
<ATTR-VALUE ITEM-ID="24">
  <COL-VALUE>course</COL-VALUE>
</ATTR-VALUE>
<ATTR-VALUE ITEM-ID="25">
  <COL-VALUE>course</COL-VALUE>
</ATTR-VALUE>
<ATTR-VALUE ITEM-ID="26">
  <COL-VALUE>course</COL-VALUE>
</ATTR-VALUE>
<ATTR-VALUE ITEM-ID="27">
  <COL-VALUE>course</COL-VALUE>
</ATTR-VALUE>
<ATTR-VALUE ITEM-ID="28">
  <COL-VALUE>course</COL-VALUE>
</ATTR-VALUE>
<ATTR-VALUE ITEM-ID="29">
  <COL-VALUE>course</COL-VALUE>
</ATTR-VALUE>
<ATTR-VALUE ITEM-ID="30">
  <COL-VALUE>course</COL-VALUE>
</ATTR-VALUE>
```

```
<ATTR-VALUE ITEM-ID="31">
  <COL-VALUE>person</COL-VALUE>
</ATTR-VALUE>
<ATTR-VALUE ITEM-ID="32">
  <COL-VALUE>person</COL-VALUE>
</ATTR-VALUE>
<ATTR-VALUE ITEM-ID="33">
  <COL-VALUE>person</COL-VALUE>
</ATTR-VALUE>
<ATTR-VALUE ITEM-ID="34">
  <COL-VALUE>person</COL-VALUE>
</ATTR-VALUE>
<ATTR-VALUE ITEM-ID="35">
  <COL-VALUE>person</COL-VALUE>
</ATTR-VALUE>
<ATTR-VALUE ITEM-ID="36">
  <COL-VALUE>person</COL-VALUE>
</ATTR-VALUE>
<ATTR-VALUE ITEM-ID="37">
  <COL-VALUE>person</COL-VALUE>
</ATTR-VALUE>
<ATTR-VALUE ITEM-ID="38">
  <COL-VALUE>person</COL-VALUE>
</ATTR-VALUE>
<ATTR-VALUE ITEM-ID="39">
  <COL-VALUE>person</COL-VALUE>
</ATTR-VALUE>
<ATTR-VALUE ITEM-ID="40">
  <COL-VALUE>person</COL-VALUE>
</ATTR-VALUE>
<ATTR-VALUE ITEM-ID="41">
  <COL-VALUE>MSc</COL-VALUE>
</ATTR-VALUE>
<ATTR-VALUE ITEM-ID="42">
  <COL-VALUE>MSc</COL-VALUE>
</ATTR-VALUE>
<ATTR-VALUE ITEM-ID="43">
  <COL-VALUE>MSc</COL-VALUE>
</ATTR-VALUE>
<ATTR-VALUE ITEM-ID="44">
  <COL-VALUE>MSc</COL-VALUE>
</ATTR-VALUE>
<ATTR-VALUE ITEM-ID="45">
  <COL-VALUE>MSc</COL-VALUE>
</ATTR-VALUE>
<ATTR-VALUE ITEM-ID="46">
  <COL-VALUE>MSc</COL-VALUE>
</ATTR-VALUE>
```

```
  <ATTR-VALUE ITEM-ID="47">
    <COL-VALUE>MSc</COL-VALUE>
  </ATTR-VALUE>
  <ATTR-VALUE ITEM-ID="48">
    <COL-VALUE>MSc</COL-VALUE>
  </ATTR-VALUE>
  <ATTR-VALUE ITEM-ID="49">
    <COL-VALUE>MSc</COL-VALUE>
  </ATTR-VALUE>
  <ATTR-VALUE ITEM-ID="50">
    <COL-VALUE>MSc</COL-VALUE>
  </ATTR-VALUE>
  <ATTR-VALUE ITEM-ID="51">
    <COL-VALUE>MSc</COL-VALUE>
  </ATTR-VALUE>
  <ATTR-VALUE ITEM-ID="52">
    <COL-VALUE>MSc</COL-VALUE>
  </ATTR-VALUE>
  <ATTR-VALUE ITEM-ID="53">
    <COL-VALUE>MSc</COL-VALUE>
  </ATTR-VALUE>
  <ATTR-VALUE ITEM-ID="54">
    <COL-VALUE>MSc</COL-VALUE>
  </ATTR-VALUE>
  <ATTR-VALUE ITEM-ID="55">
    <COL-VALUE>MSc</COL-VALUE>
  </ATTR-VALUE>
  <ATTR-VALUE ITEM-ID="56">
    <COL-VALUE>MSc</COL-VALUE>
  </ATTR-VALUE>
  <ATTR-VALUE ITEM-ID="57">
    <COL-VALUE>MSc</COL-VALUE>
  </ATTR-VALUE>
  <ATTR-VALUE ITEM-ID="58">
    <COL-VALUE>MSc</COL-VALUE>
  </ATTR-VALUE>
  <ATTR-VALUE ITEM-ID="59">
    <COL-VALUE>MSc</COL-VALUE>
  </ATTR-VALUE>
  <ATTR-VALUE ITEM-ID="60">
    <COL-VALUE>MSc</COL-VALUE>
  </ATTR-VALUE>
</ATTRIBUTE>

<ATTRIBUTE NAME="program">
  <ATTR-VALUE ITEM-ID="41">
    <COL-VALUE>MSc in Mobile and Web Computing</COL-VALUE>
  </ATTR-VALUE>
```

```
<ATTR-VALUE ITEM-ID="42">
  <COL-VALUE>MSc in ICT Systems</COL-VALUE>
</ATTR-VALUE>
<ATTR-VALUE ITEM-ID="43">
  <COL-VALUE>MSc in e-Business and Digital Marketing</COL-VALUE>
</ATTR-VALUE>
<ATTR-VALUE ITEM-ID="44">
  <COL-VALUE>Msc in Communications and Cybersecurity</COL-VALUE>
</ATTR-VALUE>
<ATTR-VALUE ITEM-ID="45">
  <COL-VALUE>MSc in Mobile and Web Computing</COL-VALUE>
</ATTR-VALUE>
<ATTR-VALUE ITEM-ID="46">
  <COL-VALUE>MSc in Mobile and Web Computing</COL-VALUE>
</ATTR-VALUE>
<ATTR-VALUE ITEM-ID="47">
  <COL-VALUE>MSc in Mobile and Web Computing</COL-VALUE>
</ATTR-VALUE>
<ATTR-VALUE ITEM-ID="48">
  <COL-VALUE>MSc in Mobile and Web Computing</COL-VALUE>
</ATTR-VALUE>
<ATTR-VALUE ITEM-ID="49">
  <COL-VALUE>MSc in ICT Systems</COL-VALUE>
</ATTR-VALUE>
<ATTR-VALUE ITEM-ID="50">
  <COL-VALUE>MSc in ICT Systems</COL-VALUE>
</ATTR-VALUE>
<ATTR-VALUE ITEM-ID="51">
  <COL-VALUE>MSc in ICT Systems</COL-VALUE>
</ATTR-VALUE>
<ATTR-VALUE ITEM-ID="52">
  <COL-VALUE>MSc in ICT Systems</COL-VALUE>
</ATTR-VALUE>
<ATTR-VALUE ITEM-ID="53">
  <COL-VALUE>MSc in e-Business and Digital Marketing</COL-VALUE>
</ATTR-VALUE>
<ATTR-VALUE ITEM-ID="54">
  <COL-VALUE>Msc in Communications and Cybersecurity</COL-VALUE>
</ATTR-VALUE>
<ATTR-VALUE ITEM-ID="55">
  <COL-VALUE>MSc in e-Business and Digital Marketing</COL-VALUE>
</ATTR-VALUE>
<ATTR-VALUE ITEM-ID="56">
  <COL-VALUE>MSc in e-Business and Digital Marketing</COL-VALUE>
</ATTR-VALUE>
<ATTR-VALUE ITEM-ID="57">
  <COL-VALUE>MSc in e-Business and Digital Marketing</COL-VALUE>
</ATTR-VALUE>
```

```
    <ATTR-VALUE ITEM-ID="58">
      <COL-VALUE>Msc in Communications and Cybersecurity</COL-VALUE>
    </ATTR-VALUE>
    <ATTR-VALUE ITEM-ID="59">
      <COL-VALUE>Msc in Communications and Cybersecurity</COL-VALUE>
    </ATTR-VALUE>
    <ATTR-VALUE ITEM-ID="60">
      <COL-VALUE>Msc in Communications and Cybersecurity</COL-VALUE>
    </ATTR-VALUE>
  </ATTRIBUTE>

  <ATTRIBUTE NAME="name">
    <ATTR-VALUE ITEM-ID="31">
      <COL-VALUE>Dr. Tjortzis</COL-VALUE>
    </ATTR-VALUE>
    <ATTR-VALUE ITEM-ID="32">
      <COL-VALUE>Dr. Berberidis</COL-VALUE>
    </ATTR-VALUE>
    <ATTR-VALUE ITEM-ID="33">
      <COL-VALUE>Dr. Gatzianas</COL-VALUE>
    </ATTR-VALUE>
    <ATTR-VALUE ITEM-ID="34">
      <COL-VALUE>Dr. Karaliopoulos</COL-VALUE>
    </ATTR-VALUE>
    <ATTR-VALUE ITEM-ID="35">
      <COL-VALUE>Dr. Kotsia</COL-VALUE>
    </ATTR-VALUE>
    <ATTR-VALUE ITEM-ID="36">
      <COL-VALUE>Dr. Ampatzoglou</COL-VALUE>
    </ATTR-VALUE>
    <ATTR-VALUE ITEM-ID="37">
      <COL-VALUE>Dr. Chatzimisios</COL-VALUE>
    </ATTR-VALUE>
    <ATTR-VALUE ITEM-ID="38">
      <COL-VALUE>Dr. Kotsia</COL-VALUE>
    </ATTR-VALUE>
    <ATTR-VALUE ITEM-ID="39">
      <COL-VALUE>Dr. Fouskas</COL-VALUE>
    </ATTR-VALUE>
    <ATTR-VALUE ITEM-ID="40">
      <COL-VALUE>Dr. Papadopoulos</COL-VALUE>
    </ATTR-VALUE>
  </ATTRIBUTE>
 </ATTRIBUTES>
</PROFDB>
```

## OrientDB Queries Yelp database

*Query #1*

```
TRAVERSE * FROM #21:0 LIMIT 20
```

*Query #2*

```
SELECT FROM business WHERE stars >= 4 LIMIT 100
```

*Query #3*

```
SELECT FROM business WHERE categories CONTAINSTEXT 'Fast Food' LIMIT
1000
```

*Query #4*

```
TRAVERSE out('reviewed_by') FROM (SELECT FROM business WHERE name
="Wendy's")
```

*Query #5*

```
SELECT FROM (TRAVERSE outV(), outE(), inV() FROM (SELECT FROM business
WHERE name ="Wendy's")) LIMIT 300
```

*Query #6*

```
TRAVERSE inE(), outV() FROM (SELECT FROM user WHERE name = 'David')
LIMIT 200
```

*Query #7*

```
TRAVERSE outV(), outE(), inV() FROM (SELECT FROM business WHERE catego-
ries CONTAINSTEXT 'Restaurants' LIMIT 10000) LIMIT 5000
```

*Query #8*

```
TRAVERSE out_reviewed_by, inV() FROM (SELECT FROM business WHERE name
containstext "Wendy's") LIMIT 10000
```

## OrientDB Queries Movies database

*Query #1*

TRAVERSE * FROM #29:0 LIMIT 100

*Query #2*

SELECT FROM Movie WHERE name CONTAINSTEXT '(1995)' LIMIT 100

*Query #3*

TRAVERSE in('is_genre') FROM (SELECT FROM Genre WHERE name = 'Fantasy') LIMIT 100

*Query #4*

SELECT FROM (TRAVERSE outV(), outE(), inV() FROM (SELECT FROM Movie WHERE name ='Cinderella III: A Twist in Time (2007)')) LIMIT 20

*Query #5*

TRAVERSE outV(), outE(), inV() FROM (SELECT FROM Movie WHERE name CONTAINSTEXT 'Batman' LIMIT 1000) LIMIT 500