Improvement of recall measure by deriving graph features for link prediction on machine learning algorithms

DISSERTATION SUBMITTED IN PART FULFILMENT OF THE REQUIREMENTS FOR THE DEGREE OF

MSc IN DATA ANALYTICS

AT DUBLIN BUSINESS SCHOOL

ZELIHA BILGE BILDIK

DECLARATION:

I declare that this dissertation that I have submitted to Dublin Business School for the award of MSc in Data Analytics is the result of my own investigations, except where otherwise stated, where it is clearly acknowledged by references. Furthermore, this work has not been submitted for any other degree.

Signed: Zeliha Bilge Bildik

Student Number: 10504293

Date: 26/08/2019

ACKNOWLEDGEMENTS:

First and foremost, I would like to express my respect and special thanks of gratitude of my supervisor Ms. Terri Hoare for her time. Her valuable guidance during my project and her extensive knowledge that was helping me to develop and complete this research.

I would like to thank my family to encourage me for my further education and supporting me with love and understanding. I would also thank to Louise and Martin Loscher for their support and guidance.

ABSTRACT:

In recent years the volume of data has increased significantly creating new challenges and opportunities in dealing with the interconnected data. Although new technologies enable the processing of high volumes of information, it is still challenging to find the relationships within the data that realise the anticipated business value. Graph analysis is becoming increasingly important to find the insights from connected data and to leverage machine learning outcomes. This thesis presents graph analytics applied on the leading ACID compliant graph dbms Neo4j to derive the features to improve on the prediction of recommender algorithms. The research uses the Movielens dataset for benchmarking purposes. Python is used for building the data pipeline using embedded cypher and python machine learning libraries.

The research demonstrates the effectiveness of link prediction as a method for derivation of the features for machine learning. The resultant improvements in recall are demonstrated.

Keywords: Graph Databases, Neo4j, Graph Analytics, Feature Selection, Link Prediction, Movies

Table of Contents

List of Figures:

Chapter 1

Introduction:

Nowadays, data is seen as new oil in the digital world in order to improve business services and take competitive advantage in the market. However, data itself does not provide an advantage unless it is converted to the relevant information. Data analysis and decision-making process are emerging with big data technologies which allows us to process a high volume of data where the traditional data base systems would find it more difficult to operate.

Increases in volume has made finding the relationships within the data more and more important. Use of correct big data technologies and data analytics techniques help to find valuable information and insights. These insights can have a huge impact for today's business. It enables innovations, as well as the development of new services, new markets and new products and the improvement of them. Therefore, organizations who can effectively process their data to reach relevant information for the decision-making process can then take advantage of competitive markets. Companies such as google, amazon and facebook who use information and technologies in the appropriate context are now leading the market.

Big data defines as a volume, velocity, veracity and variety. Big data also related with the connectivity of the data. Because of that, the graph technologies have significant importance in this era that provide easy analysis on interconnected data.

Graph Analytics are shown among 2019 top ten data and analytics technology trend on Gartner. ('Newsroom', 2019)

1.1 Research Questions and Research Objectives

Research Question: Observing how graph features improve machine learning results.

Research problem: Predicting new links, future collaboration, between users who will watch the same movie in common.

Research Objective: Graph algorithms and NEO4J as a graph database will be examined by finding the answers to the following questions. How to model data and set the relationships in Neo4j, how to find similarities between the data by using graph algorithms, how to enhance machine learning by using graph features in link prediction problems and how to implement graph algorithms by using python programming language.

Hypothesis: Graph features improve the machine learning algorithms performance.

Research sources and tools: MovieLens Dataset, graph database NEO4J, python and cypher query languages.

1.2 Motivation

I have worked as a data professional for 12 years in different areas in data management. Data volumes and varieties have risen all that years and the business needs have changed. In today's digital world, summarizing the data, getting big picture and generalize it for the domains not enough for the expectations and data driven approaches. The use of both structural and non-structural data has become important for detailed analysis to find consumable and valuable information. In addition, new technologies provide analysis capabilities on not only the past and present (real time) activities, but also future activities. I have become quite interested in big data and analytics area and their abilities especially in transactional data because there is a precious information behind the transactions. I have interested in finding patterns, predict the next actions, finding behavioural relationships, correlations etc. within the data. Traditional databases do not enough to analyse complex relationships within the high volume of data in a way that graph databases do. Therefore, I tend to learn graph databases more deeply by combining their potential on machine learning process.

In addition, I wanted to use my skills and background to emphasize data preparation phase by finding out, if the graph database would have any contribution to shorten this phase. Because this is the most time-consuming stage in the analytical projects. Besides, I tend to work on

methodology to present steps on analytical studies which can provide frame for similar studies in this area.

1.3 Scope

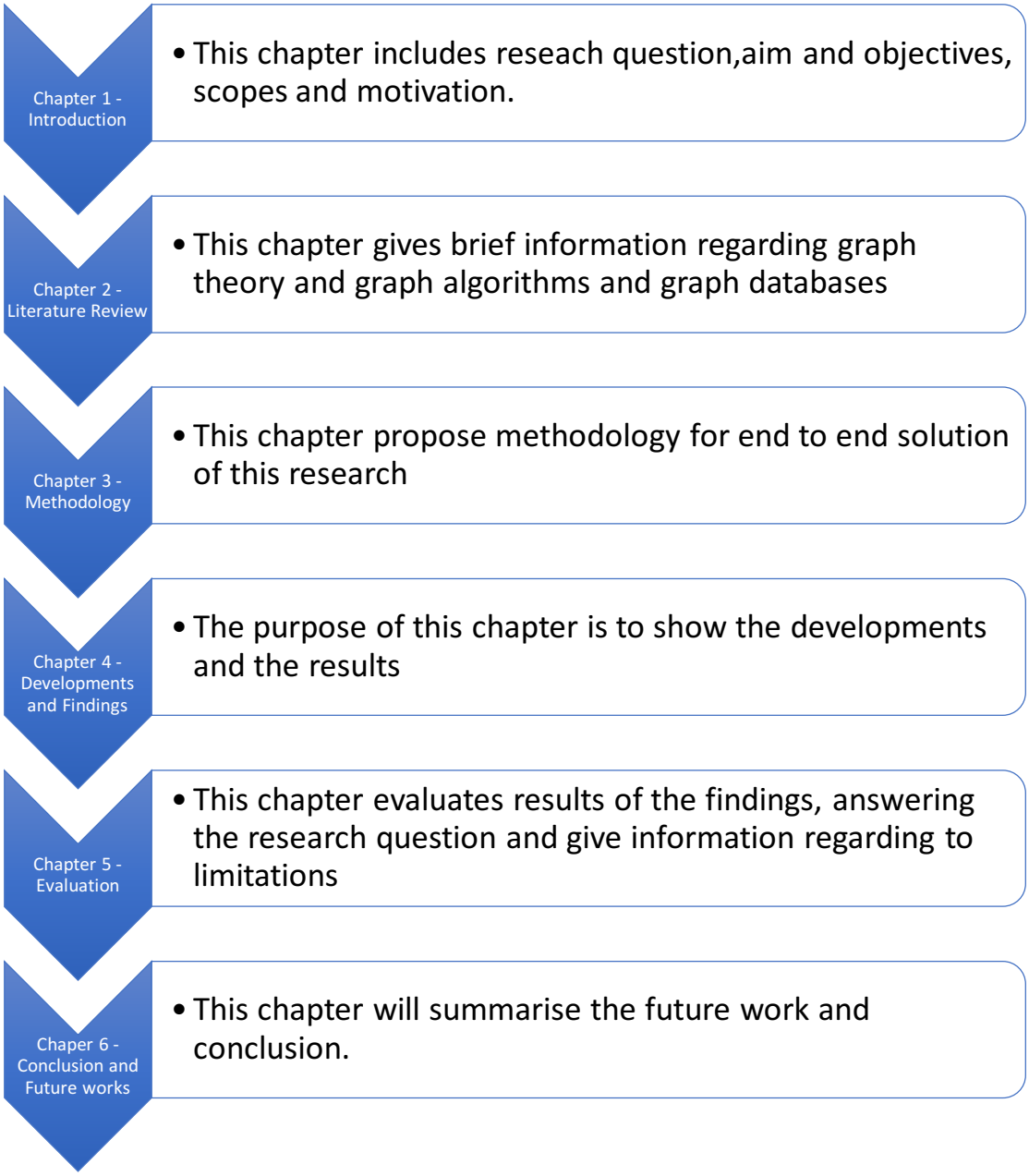
The scope of this research is to build end to end solutions for link prediction problems with graph feature engineering by combining CIRSP-DM methodologies with personal experience.

1.4 Contributions

The contribution of this research has three steps. First, Neo4j database has been created as a source database, the suitable environmental setting has been done for the connection from python environment. Then dataset has been loaded and the graph modelling has been applied to the data. Second, graph analytics and graph features has been studied in depth and efficient solution for link prediction problem has been found and implemented. The effect of the graph features on machine learning problem has been illustrated. Third, methodology has been studied during the implementation in order to propose new steps for analytic studies.

1.5 Dissertation Road Map

The rest of the research is designed as follows. Chapter 2 introduces the background for the research, theorems, descriptions. Chapter 3 is the followed methodology throughout the development. Chapter 4 describes the design and implementation of the solution, while Chapter 5 conclusion and further study in this area.



Chapter 2

Literature Review

Introduction:

The following sections will provide information by considering literature review, graph theory, basic definitions for graph algorithms. Because research area on graph analytics and algorithms could be considered as new, very few research paper was founded. Internet sources and books were also used in this research.

(Needham and Hodler, 2019), graph analytics book describes graph algorithms and their basic definitions. It not only provides organized detailed information about graph algorithms and graph analytics but also give information about hands on development with examples and descriptions. This research was followed (Needham and Hodler, 2019), graph analytics book for algorithms descriptions and developments.

Mutlu and Oghaz (2019) explains the general techniques for link prediction. This article presents feature extraction methods form on similarity measures, maximum likelihood and probabilistic methods and graph learning. Detailed information is provided by splitting models into two categories one is Feature Extraction Methods the other is Feature Learning Methods. "The first branch consists of models studying network features through Similarity Based Methods, Likelihood Based Methods, and Probabilistic Methods. The second branch contains techniques which apply a node embedding to the graph and learn the graph representations. These models include Matrix Factorization Methods, Random Walk Based Methods, and Neural Network Based Methods. All of the studied techniques provide features which can be fed to supervised/unsupervised machine learning algorithms to approach the link prediction problem through a clustering or a classification task." (Mutlu and Oghaz, 2019)

Analysis by Yang, Lichtenwalter and Chawla (2015, pp. 751- 782) separate link prediction problem into two categories first one is prediction of links which will be the new link in the network. These can be applied for future collaboration and future friendship problems. Second one is inferring the missing links from the observation of the network. This method used to clarify lost and hidden links.

Gupta et al. (2015) analyse various link prediction algorithms based on performance metrics accuracy, precision, specificity and sensitivity. It presents link prediction algorithms with three approaches, Local Similarity Features (Common Neighbors, Jaccard's Coefficient, Adamic-Adar, PreferentialAttachment, ResourceAllocation), Global Similarity and Feature Vector.

Research Battaglia et al. (2018) represent graph networks and its importance for human – like intelligence in further AI studies. The paper shows three principles on graph networks which are flexible representations, configurable within-block structure and composable multi-block architectures. Flexible representation uses graph notations, edges nodes and the output would be used in graph networks. Battaglia et al. (2018) states that "Recent advances in AI, propelled by deep learning, have been transformative across many important domains. Despite this, a vast gap between human and machine intelligence remains, especially with respect to efficient, generalizable learning."

Martinez, Berzal and Cubero (2019) review techniques for link prediction problem in their survey "A Survey of Link Prediction in Complex Networks". This article gives information about classifier based methods and pointed out the link prediction problem as relatively new research area.

2.1 Graph Theory History and Overview:

Graphs were first introduced in 1736, when Leonhard Euler, Swiss mathematician, solved the "Seven Bridges of Königsberg" problem. "The question presented to Euler was: "Was it

possible to take a walk through the town in such a way as to cross over every bridge once, and only once (known as a Euler walk)?" (Najera, 2018)

Euler set the preliminary work for graph theory and its mathematics by providing first visual representation with the insight that only the connections themselves were relevant. Figure 1-1 shows Euler's progression with one of his original sketches, from the paper "Solutio problematis ad geometriam situs pertinentis". (Needham and Hodler, 2019, p. 2)

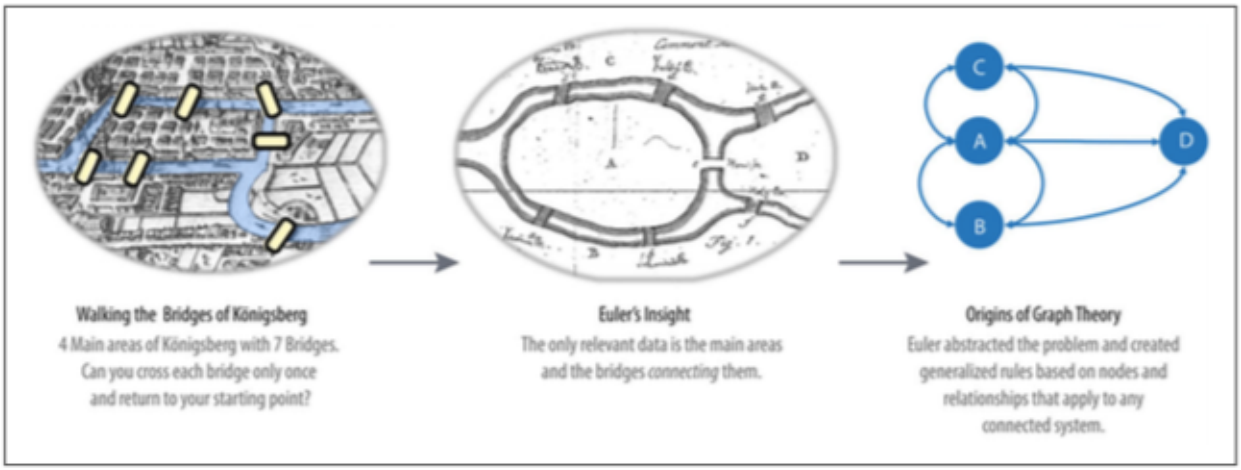


Figure 1-1. The origins of graph theory. The city of Königsberg included two large islands connected to each other and the two mainland portions of the city by seven bridges. The puzzle was to create a walk through the city, crossing each bridge once and only once.

Figure 2.1 The origins of graph theory (Needham and Hodler, 2019, p. 2)

Graph Theory is described as the study of **relationships**. Graph theory is helpful approach for dynamic systems that provides solutions for wide variety of problems by simplifying the connections and its relationships between interconnected data. Graphs are represented by nodes or vertices and links between the nodes are known as relationships or edges. (Needham and Hodler, 2019, p. 2)

Mathematical expressions defined for the graph by Wilson (2010, p.8) as:

> A simple graph G consists of a non-empty finite set V(G) of elements called vertices (or nodes), and a finite set E(G) of distinct unordered pairs of distinct elements of V(G) called edges. V(G)the vertex set and E(G) the edge set of G. An edge{v,w} is said to join the vertices v and w, and is usually abbreviated to vw.

Two nodes(vertices) v and w of a graph G are adjacent if there is an edge of nodes(vertices) joining them. Similarly, two distinct nodes are adjacent if they have a link in common. (Wilson, 2010, p.8). "A subgraph of a graph G is a graph, each of whose nodes belongs to V(G) and each of whose relationships belongs to E(G)". (Wilson, 2010, p.8)

Matrix representations: "Although it is convenient to represent a graph by a diagram of points joined by lines, such a representation may be unsuitable if we wish to store a large graph in a computer. One way of storing a simple graph is by listing the vertices adjacent to each vertex of the graph." (Wilson, 2010, p.14). The matrix representation is described by Wilson (2010, p.14) as:

> Other useful representations involve matrices. If G is a graph with vertices labelled {1, 2, ...,n}, its adjacency matrix A is the n x n matrix whose ij-th entry is the number of edges joining vertex i and vertex j. If, in addition, the edges are labelled {1, 2,..., m}, its incidence matrix M is the n x m matrix whose ij-th entry is 1if vertex i is incident to edge j, and 0 otherwise.

Figure 2.2 Matrix representation of graph, based on (Wilson, 2010, p.14)

2.2 Graph Types and Structures:

In graph theory, graph types are represented with their relationships. A simple graph where nodes have one relationship. If nodes have multiple relationships, it is called multigraph. Nodes can have relationships with themselves in that case it is named as pseudograph. (Needham and Hodler, 2019, p.16-17)

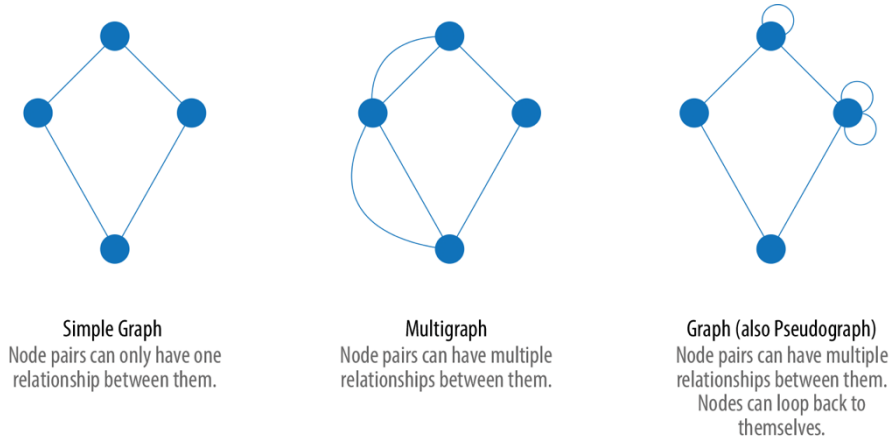


Figure 2.3 Basic Graph Types, based on Needham and Hodler, 2019, p.17.

2.3 Common Graph attributes:

2.3.1 Connected Versus Disconnected Graphs

Connected graph has a path between all nodes. Unlike connected graph, disconnected graphs do not have path between all nodes, if the nodes in those independent paths are connected, they are called components (clusters). (Needham and Hodler, 2019, p.19)

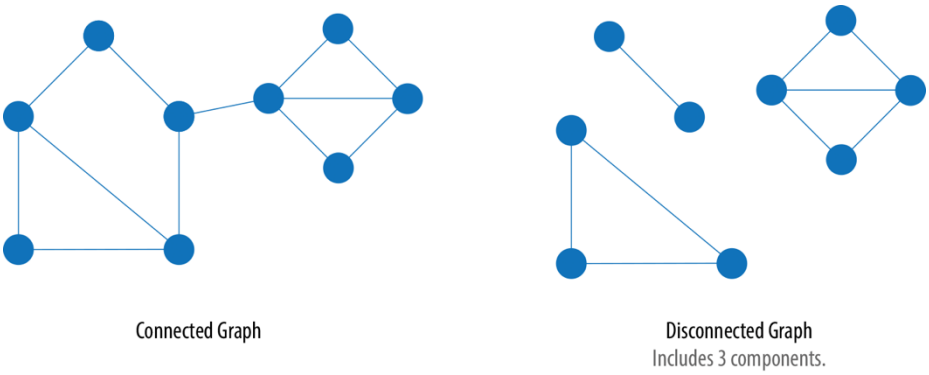


Figure 2.4 Connected vs Disconnected Graphs, based on Needham and Hodler, 2019, p.19.

Detailed explanation with regarding connected graphs given by Wilson (2010, p.26) as:

> Given a graph G, a walk in G is a finite sequence of edges of the form $v_0\ v_1$, $v_1\ v_2$, …, $v_{m-1}\ v_m$, also denoted by $v_0 —> v_1 —> v_2 —> v_m$, in which any two consecutive edges are adjacent or identical. Such a walk determines a sequence of vertices $v_0, v_1 . . ., v_m$. We call v_0 the **initial** vertex and v_m the final vertex of the walk, and speak of a walk from v_0 **to** v_m. The number of edges in a walk is called its length.

2.3.2 Unweighted Graphs Versus Weighted Graphs

If any of values or attributes are assigned to their nodes or relationships of graph such as cost, time, distance, capacity this graph is called as weighted graph. Otherwise graph is named as unweighted graph. Figure 2-4 shows weighted and unweighted graph representation.

Figure 2.5 Weighted graphs can hold values on relationships or nodes, (Needham and Hodler, 2019, p.20).

2.3.3 Undirected Graphs Versus Directed Graphs

Unless the directions are not defined for relationships, undirected graphs, that is considered as bidirectional. Graph is directed graph, if the directions of relationships are pointed. Directions are important for data flow and connectivity of the information. (Needham and Hodler, 2019, p.21)

2.3.4 Acyclic Graphs Versus Cyclic Graphs

Cycles are basics for many graph algorithms. Cyclic graph starts and ends from the same node, unlike cyclic graph acyclic graph has not cycles starting and ending nodes are different. Cycles exist in directed and undirected graph types. In directed graphs path follows the relationship direction. (Needham and Hodler, 2019, p.21)

2.3 Graph Analytics and Algorithms:

According to Gartner, "Graph analytics is a set of analytic techniques that allows for the exploration of relationships between entities of interest such as organizations, people and transactions." ('Newsroom', 2019)

The main idea behind the graph analytics is use of graph approach to analyse connected data and find hidden patterns. Graph algorithm's mathematical calculations specifically built to perform on relationships, graph algorithms are one of the most important approach to analysing the connected data. Besides graph algorithms are used for the graph analytics especially data science purposes. (Needham and Hodler, 2019, p.21)

Graphs help uncover how tiny interactions and dynamics cause to large effect on structures. They come together the micro and macro scales by showing exactly which entities are interacting within large structures. These interactions help to predict missing links and behaviours. (Needham and Hodler, 2019, p.21)

According to Needham and Hodler(2019) Graph algorithms are used to find relevant information from connected data. Wide range of graph representation within real-world systems from protein interactions to social networks, from communication systems to power grids, and from retail experiences to Mars mission planning. One of the biggest advantage that graphs give information about the network and its interactions which would be useful insights and guide for the new developments. (Needham and Hodler, 2019, p.21)

Graph Analytics Use Cases shown by Needham and Hodler(2019) as:

- Investigate the route of a disease or a cascading transport failure.
- Uncover the most vulnerable, or damaging, components in a network attack.
- Identify the least costly or fastest way to route information or resources.
- Predict missing links in your data.
- Locate direct and indirect influence in a complex system.

- Discover unseen hierarchies and dependencies.
- Forecast whether groups will merge or break apart.
- Find bottlenecks or who has the power to deny/provide more resources.
- Reveal communities based on behaviour for personalized recommendations.
- Reduce false positives in fraud and anomaly detection.
- Extract more predictive features for machine learning.

2.4 Types of Graph Algorithms

2.4.1 Pathfinding and Graph Search Algorithms

In a given graph (network) pathfinding algorithms basically examine paths between nodes by moving through the relationships(links). Different algorithms can be used under the pathfinding in order to find ways to reach the purpose node. (Needham and Hodler, 2019, p.39)

Breadth-First Search (BFS): It starts at the chosen node and reach the final wanted node by exploring the all of its nearest neighbors and then their sublevel neighbors. (Needham and Hodler, 2019, p.45)

Depth-First Search (DFS): It starts from selected node and pick one of its neighbors and moving through as far as it can before back tracking. (Needham and Hodler, 2019, p.45)

Shortest Path: The Shortest Path algorithm finds the shortest path between two chosen nodes. (Needham and Hodler, 2019, p.49)

All Pairs Shortest Path: Computes the shortest path between all pairs of nodes in the graph

Single Source Shortest Path: Computes the shortest path between a single root node and all other nodes. This algorithm cannot use with negative weights of relationships (Needham and Hodler, 2019, p.60)

Minimum Spanning Tree: Computes the path in a connected tree structure with the smallest score for visiting all nodes. (Needham and Hodler, 2019, p.71). MST algorithm works well

when the relationships have different weights. ('The Neo4j Graph Algorithms User Guide v3.5', no date)

Random Walk: Returns a list of nodes along a path of specified size by randomly choosing relationships to traverse.

Random walk cannot be applied under which conditions are described by Needham and Hodler (2019) as below:

- Dead-ends occur when pages have no out-links. In this case, the random walk will abort and a path containing only the first node will be returned. This problem can be avoided by passing the direction: BOTH parameter, so that the random walk will traverse relationships in both directions
- If there are no links from within a group of pages to outside of the group, then the group is considered a spider trap. Random walks starting from any of the nodes in that group will only traverse to the others in the group - our implementation of the algorithm doesn't allow a random walk to jump to non-neighboring nodes.
- Sinks can occur when a network of links form an infinite cycle.

2.4.2 Centrality Algorithms

Centrality algorithms are used in order to understand the roles of specific nodes in a graph and their impact on that graph. They analyse the most crucial nodes and helps to understand group dynamics such as credibility, accessibility, the speed at which things spread, and bridges between groups. (Needham and Hodler, 2019, p.77-99)

Degree Centrality: Degree centrality is used for finding popular nodes in a network by counting the number of incoming and outgoing relationships from a graph. (Needham and Hodler, 2019, p.77-99)

Closeness Centrality: Closeness centrality finds nodes which are capable to spread the information through a graph. "The closeness centrality of a node measures its average farness

(inverse distance) to all other nodes. Nodes with a high closeness score have the shortest distances to all other nodes." ('The Neo4j Graph Algorithms User Guide v3.5', no date)

Betweenness Centrality: Purpose of Betweenness centrality is detecting the amount of influence a node. It is often used to determine nodes that serve as a bridge from one part of a network to another. (Needham and Hodler, 2019, p.77-99)

PageRank: PageRank is the best known of the centrality algorithms. "PageRank is named after Google cofounder Larry Page, who created it to rank web- sites in Google's search results. PageRank measures the number and quality of incoming relationships to a node to determine an estimation of how important that node is. "(Needham and Hodler, 2019, p.77-99)

2.4.3 Community Detection Algorithms

In order to evaluate group behaviour in networks community formation is common in all types of graph. Finding communities helps to determine similar behaviours or preferences, find nested relationships. In group structure nodes have more relationships within the same group than nodes outside the groups. (Needham and Hodler, 2019, p.109-133)

Triangle Count and Clustering Coefficient

The goal of the Clustering Coefficient algorithm is to measure the closeness in which nodes come together. Triangle Count is used in its calculations to give a ratio of existing triangles to possible relationships. (Needham and Hodler, 2019, p.109-133). Needham and Hodler indicates that "Triangle Count determines the number of triangles passing through each node in the graph. A triangle is a set of three nodes, where each node has a relationship to all other nodes." There are two types clustering coefficient which are local and global. "The global clustering coefficient is the normalized sum of the local clustering coefficients." (Needham and Hodler, 2019, p.109-133)

The clustering coefficient of a node can be found by multiplying the number of triangles passing through the node by two and then diving that by the maximum number of relationships in the

group, which is always the degree of that node, minus one. Examples of different triangles and clustering coefficients for a node with five relationships are portrayed in Figure. (Needham and Hodler, 2019, p.109-133)

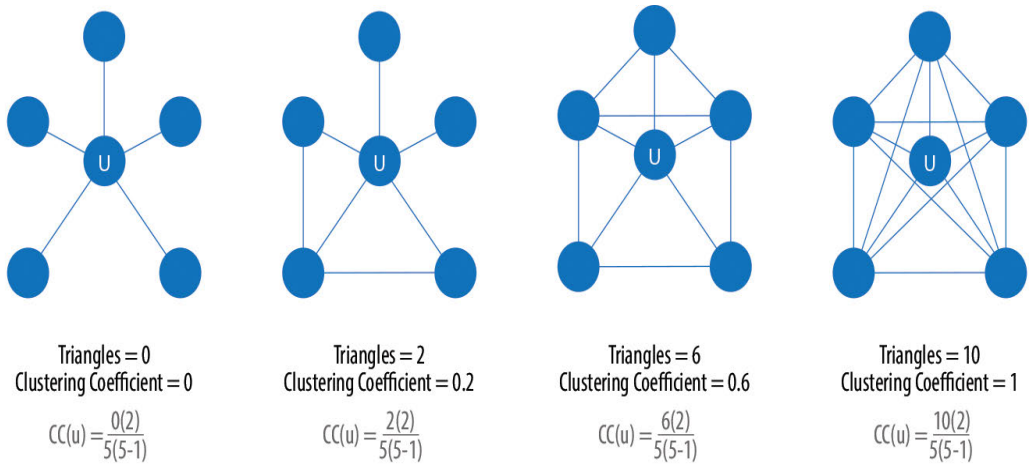


Figure 2.6 Triangle counts and clustering coefficients for node u (Needham and Hodler, 2019, p.115).

Formula for clustering coefficient is:

$$CC(u) = \frac{2R_u}{k_u(k_u - 1)}$$

where: u is a node.

R(u) is the number of relationships through the neighbors of u (this can be obtained by using the number of triangles passing through u).

k(u) is the degree of u. (Needham and Hodler, 2019, p.109-133)

The Louvain algorithm

The Louvain Modularity algorithm compares community density in order to find clusters and assigns nodes to different clusters. (Needham and Hodler, 2019, p.109-133) "Louvain quantifies how well a node is assigned to a group by looking at the density of connections within a cluster in comparison to an average or random sample. This measure of community assignment is called modularity." ('The Neo4j Graph Algorithms User Guide v3.5', no date)

Label Propagation

The Label Propagation algorithm (LPA) is used for examining the communities in a graph. The groups are selected by nodes depending on their direct connections. One of the use cases for label propagation described in Neo4j documentation as "Label propagation has been used to assign polarity of tweets, as a part of semantic analysis which uses seed labels from a classifier trained to detect positive and negative emoticons in combination with Twitter follower graph." ('The Neo4j Graph Algorithms User Guide v3.5', no date)

2.4.3 Link Prediction Algorithms

In this research three link prediction algorithms: common neighbors, preferential attachment and total neighbors, scores were used. The link prediction algorithms basically work on finding the similarity and closeness scores between the nodes. If these scores are high future relationships are most likely to occur.

The Common Neighbors algorithm

In Common neighbors the idea is two nodes which have not got a direct relationship but they do have a common relationship with one other node, are most likely have a link between them. It is calculated by using the below the formula:

$$CN(x, y) = |N(x, y) \cap N(y)|$$

where N(x) is the set of nodes adjacent to node x, and N(y) is the set of nodes adjacent to node y. An equivalent of 0 shows that two nodes are not close, while higher values shows nodes are closer. ('The Neo4j Graph Algorithms User Guide v3.5', no date)

The Preferential Attachment algorithm

Preferential Attachment is a score used to calculate the closeness of nodes, depend on their shared neighbors. If a node is more connected to the other nodes, the probability of having new links is higher.

It is calculated by using the below the formula:

$$PA(x, y) = |N(x)| * |N(y)|$$

where N(u) is the set of nodes adjacent to u. An equivalent of 0 shows that two nodes are not close, while higher values shows that nodes are closer. ('The Neo4j Graph Algorithms User Guide v3.5', no date)

The Total Neighbors algorithm

Total Neighbors determines the closeness of nodes by calculating the number of unique neighbors that they have. If a node is more connected to the other nodes, the probability of having new links is higher. Total Neighbors is calculated by using the below the formula:

$$TN(x, y) = |N(x) \cup N(y)|$$

where N(x) is the set of nodes adjacent to x, and N(y) is the set of nodes adjacent to y. An equivalent of 0 shows that two nodes are not close, while higher values shows that nodes are closer. ('The Neo4j Graph Algorithms User Guide v3.5', no date)

2.5 Graph Databases:

Last few decades increase in volume, velocity and variety of the data result in new storage requirements that relational database management systems not easy to operate with regarding to cost, performance and schema structures. As a result, nosql databases has been becoming more popular to manage and operate not only high volume but also different variety of the data such as structural and unstructural data.

Basically, graph databases are in NOSQL (not only sql) database family which store data in graph structure (node and relationships). Unlike the other databases relationships are the main

points for graph databases. Graph databases generally used for transactional systems(OLTP), therefore CRUD operations and ACID compliance is crucial. Graph structure use a model in a visual way that make easier to understand business domain. Graph technologies is differentiated by their storage and processing engine as well as their graph model. Most commonly used graph models are, property graphs, hypergraphs, and triples. (Robinson, Webber and Eifrem, 2015, p. 205)

Hypergraphs: Unlike the property graph model, in which a relationship to have only one start node and end node, the hypergraph can connect any number of nodes. If the domain has many-to-many relationship hyper graphs are useful. (Robinson, Webber and Eifrem, 2015, p. 209)

Triples: Triple stores data in a format known as a triple. Triples consist of a subject-predicate-object data structure. Triples do not provide index-free adjacency and the storage engines are not optimized to store property graph. (Robinson, Webber and Eifrem, 2015, p. 209)

Property graph: A property graph characteristics explained by Robinson, Webber and Eifrem (2015, p.209) as:

- It contains nodes and relationships.
- Nodes contain properties (key-value pairs).
- Nodes can be labeled with one or more labels.
- Relationships are named and directed, and always have a start and end node.
- Relationships can also contain properties

This research uses Neo4j as a graph database which is ACID-compliant property graph. Visual representation of graph model shown in the figure. (‘What is a Graph Database?’, no date)

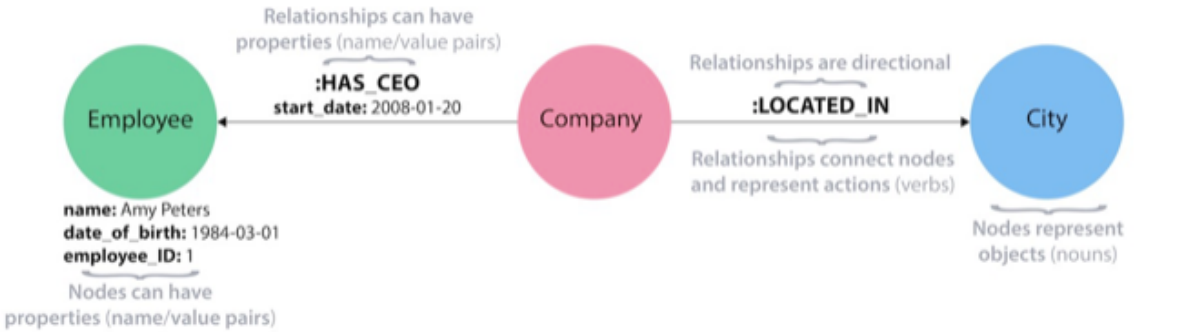


Figure 2.7 property graph illustration on neo4j database based on ('What is a Graph Database?', no date)

2.6 Machine Learning

Machine learning is a discipline based on the idea that machines can learn from the data. This learning process uses the mathematical models, statistical algorithms and methods. Machine learning process uses data which has been split into test and train set to build a model. To train the models, the training dataset is used.

Machine learning is a continuous and at the same time iterative process and can be classified into three categories: supervised learning, unsupervised learning and reinforcement learning.

Supervised and Unsupervised Learning:

Supervised learning tries to figure out a function based on labeled training data to find unlabeled data. Output variables are predicted by input data by this function. Models are developed from the training data. Relationships between input and output data sets are generalized by the model. This generalized model is used for the data where the output is unknown. (Kotu and Deshpande, 2014)

Supervised learning can be classified as a regression and classification problem. Regression is used when the output or target variable is numeric whereas classification is used when the output variable is categorical. Algorithm selection can be made by using this information. KNN,

decision trees, naïve bayes, artificial neural networks, support vector machine and ensemble methods algorithms can be used for the solution of classification problems. Linear and logistic regression can be used to provide resolution of regression problems.

Unlike supervised learning, unsupervised learning process is used on unlabeled data. “In unsupervised data science, there are no output variables to predict. The objective of this class of data science techniques, is to find patterns in data based on the relationship between data points themselves.” (Kotu and Deshpande, 2014)

Unsupervised learning can be classified as clustering and association problems. K- means and apriori algorithms are most commonly used in unsupervised learning process.

Chapter 3

Methodology

3.1 Introduction:

Generally, data projects are back and forth projects therefore, working with data projects requires full attention and collaboration from beginning to end. Methodologically, the first step is clarification of the problem definition and problem analysis, after that the purpose of the project would become clearer. In addition to this, the breaking down of goals can allow the results to be used across the entire business. Taking into account of technical and business goals helps achieve reusable results. Data projects are generally long-standing therefore, instead of focusing on just one big goal, breaking it down and separating it into achievable short-term purposes would illustrate the stage of the development of the project more clearly. This research combines CRISP-DM and personal experience.

The article "The CRISP-DM Model: The New Blueprint for Data Mining" by Shearer (2000) addresses how data mining projects should consider steps of the CRISP-DM (Cross-Industry Standard process for Data Mining) method as an industry standard by giving detailed processes definitions and giving examples about each phase of the method which is combined together by the best practices without any tools, services and industry dependencies. CRISP- DM is intended to provide guidance for any data mining projects. Because the volume of the data is getting higher every second, new technology developments have been done on tools, so new tools come into the market and some have become obsolete. However, the methodologies and guidance does not change that much. Even though the article was written in 2000, the phases are still applicable for today's big data mining projects. According to Mariscal, Marbán and Fernández, (2010, pp.137- 166) "One important factor of CRISP-DM success is the fact that CRISP-DM is industry-, tool- and application- neutral." The CRISP-DM methodology organizes analytical frame work very logical way. The phases of the CRISP-DM methodology are as follows:

Business Understanding

The first phase of the CRISP-DM methodology is business understanding which is the very crucial part for not only data mining projects but also any of other data projects. Business understanding has sub steps for clarifying the project objectives, business stand points to convert them into the data mining problem definition. These are the determinations of the business objectives, assessment of the situation, the determination of the mining goal, and the production of the final project plan. All requirements, risks and success criteria as well as the entire project plan are defined in this phase which is reasonable for clarifying the data mining project and objectives. At the end, well defined, clear objectives and the success criteria creates basics of the data mining projects for the further developments. (Shearer, 2000, p. 14)

Data Understanding

Data understanding phase is defined as a second phase in the CIRSP-DM process methodology. It consists of the initial data collection, the data description, the data exploration and the data quality verifications steps. First hypothesis and data quality reports is formed in this phase which gives important insights for the following phases. According to the quality report it can be possible to turn back to business understanding phase and make reviews on the goals and some of the criteria. (Shearer, 2000, p. 15) Although data exploration step is defined in the data understanding phase, Today's technology provides many tools for data exploration. Therefore, it could be done in the business understanding phase. Because this step directly effects all the project milestones, by using exploration tools, early detection of the possible data issues could be discovered and provide very useful insights about the project plan.

Data Preparation

Third phase of the CRISP-DM is data preparation, this has five sub steps which are the selection, the cleansing, the construction, the integration and the formatting of the data. In this phase, importance of the technology selection in the business understanding phase plays a big role to solve big data issues such as, Volume and Scalability, Miss-Handling of Big Data, Privacy and Security, Speed and Velocity, Heterogeneity of Data (Asha, 2016, p.29). In addition, this phase forms a basis for the following phase (data modelling). Data accuracy and feature selection directly effects the final result of the data modeling. The data volume related performance problems can be solved in the integration step by combining tables or

summarizing the data. Additionally, derived attributes, data transformation (e.g instead of yes or no 1 or 0) which is required for algorithms or modelling tools is to be done in this phase. It is pointed out by Shearer (2000, p. 17) that 50-70 percent of the time and effort of the data mining project is consumed in this phase. This phase could be considered as a hub for an entire project life cycle that possible issues are collected and solved, therefore the outcome of this phase is not only crucial for further phases but also previous phases.

Modelling

The selection of the modelling technique, the generation of the test design, the creation and the assessment of models is done in this phase. This phase has no certain modelling techniques because several methods can be applied for the same data mining problem. It may be possible to return back to data preparation and business understanding steps. According to Moro, Laureano, Cortez "If the obtained model is not good enough for use to support business, then a new iteration for the CRISP-DM is defined." (Moro,S. and Laureano, R., M. S., Cortez P., 2011, p.2). After building a model, empirical testing is done to evaluate the strength of the model. It should be considered that the importance of the cross functional teams and back and forth iteration come up in this step as well. Possible model outcomes and issues could be reviewed before waiting for the assessment step with cross functional teams.

Evaluation

The modelling phase is followed by the evaluation phase which has the sub steps of evaluation of results, the process review and the determination of next steps. Final decision of the project is decided in this phase by determining whether the model meets the business objectives. This phase is completed by considering the initial business success criteria, decision for the next steps whether to go back to initial steps or continue to deployment of the model. (Shearer, 2000, p. 17)

Deployment

The key steps defined here as deployment plan, monitoring and maintenance plan, final report production, and project review. (Shearer, 2000, p. 18) The general purpose of this phase is to transfer the results to daily operations. Depending on the business requirements final outcome might be report, or automation of the data mining projects. The importance of this phase is knowledge sharing, lessons learned with identifying the bottlenecks which are useful

information for new projects.

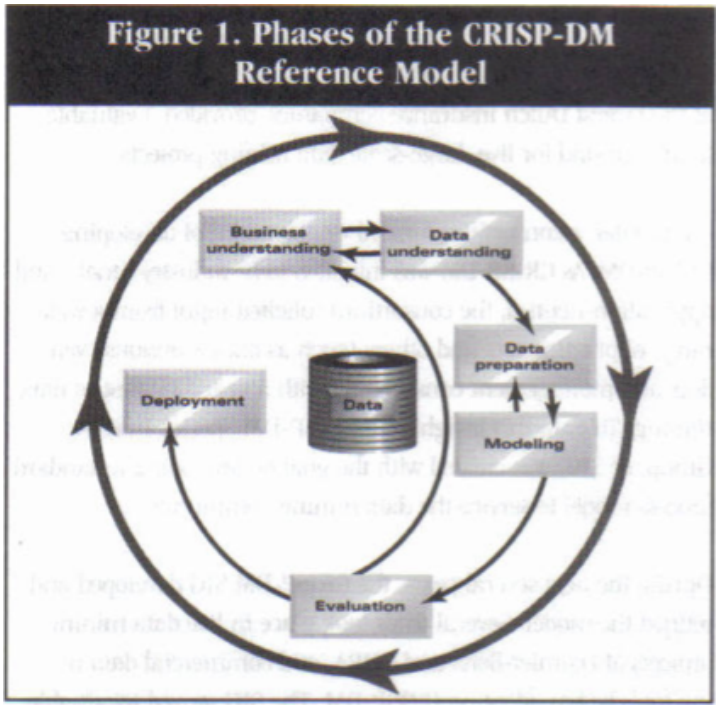


Figure 3.1 "The CRISP-DM Model: The New Blueprint for Data Mining" by Shearer (2000)

CRISP-DM has many back and forth iterations until the deployment stage ıs reached to achieve the desired business goal. To become more adaptable for a fast-paced environment, methodologies should be combined with the new principles. This section presents an overview of the research methodology used for the artefact development of this research.

3.1 Identifying the Business Goal

Clear understanding of the business problems and requirements are the first step for every project. They provide guidelines for the research or projects. The goal of this research is to show the effect of graph features on machine learning problems by predicting future interaction of users for the movies. From this interaction finding the possibilities of which users would rate or watch an upcoming movie.

3.2 Identify Analytical Framework:

Before starting the development process an analytical framework must be determined for this research that which of the learning method will be used for this business problem. It should be decided in this phase whether the problem is supervised or unsupervised. This research pointed

supervised learning and in this research random forest classifier was selected.

3.2.1 Identify, Collect and Analyse Data

In this thesis, MovieLens dataset was used with user ratings. Dataset was reviewed to understand structure and relationships in order to model it in neo4j database. The data was loaded to neo4j database which was used as the source database for other phases. Relationships and some graph queries was applied to data.

3.2.2 Identifying the available tools and programming languages

Under this phase, graph database neo4j with cypher and python programming language was examined. Graph database neo4j has many algorithms and documentation to enhance machine learning and cypher is a query language for graph database. Python has many packages for data processing and data analysis and has a direct connection to the Neo4j database by using the py2neo package.

3.3 Break down the goals

Besides the business goals, it was necessary to set technical goals in order to complete this research, such as learning graph technologies, neo4j, cypher and python, graph analytics and learning feature engineering on graph.

3.4 Prepare prototype for end to end process

Preparing prototype is useful to find undetermined problems during the implementation. Because of resource limitations on used computer for this thesis, it was needed to compute end to end solution based on sample data. Similar to the CRISP-DM modelling phase iterations were done in this stage. Data was labelled and used as a source for random forest classifier. Graph features were added one by one to evaluate the effect and resulting outcome of each feature on the results.

3.4.1 Review goals, review risks, action plans

Prototyping of end to end process with real development codes provide big picture for entire deployment. Resource allocation, infrastructure design and automation process would be mapped easily as a result of the prototyping process. Undetermined risks and possible actions would be addressed in this phase. In addition, are we close the goals or what actions can be set

to reach the goals can be evaluated in this review.

In this thesis user-user relationships and labelling the data were the main importance to predict future possible collaboration of users. During the prototyping stage, it was seen that after adding new relations and try to label them by using whole dataset was not executed because of the memory requirements. New action plan was set for development that the data was sampled to smaller size to complete the development process. Then, to make sure about the stability of the model, data size was increased until reaching the limiting point of the computer.

3.5 Apply solution to the entire dataset present the findings, set future works

Because of the resource limit on personal computer, entire data set could not be used. Instead, data size was increased experimentally and all codes which are developed from prototyping stage applied to find limit of the computer. This process was also used for the proof for stability of the model because reasonable results are observed after each of the execution. All results were added to the artefacts. Setting the new goals and future works would help for reusability and automation of the results.

The overall methodology was used in this research shown in below the figure:

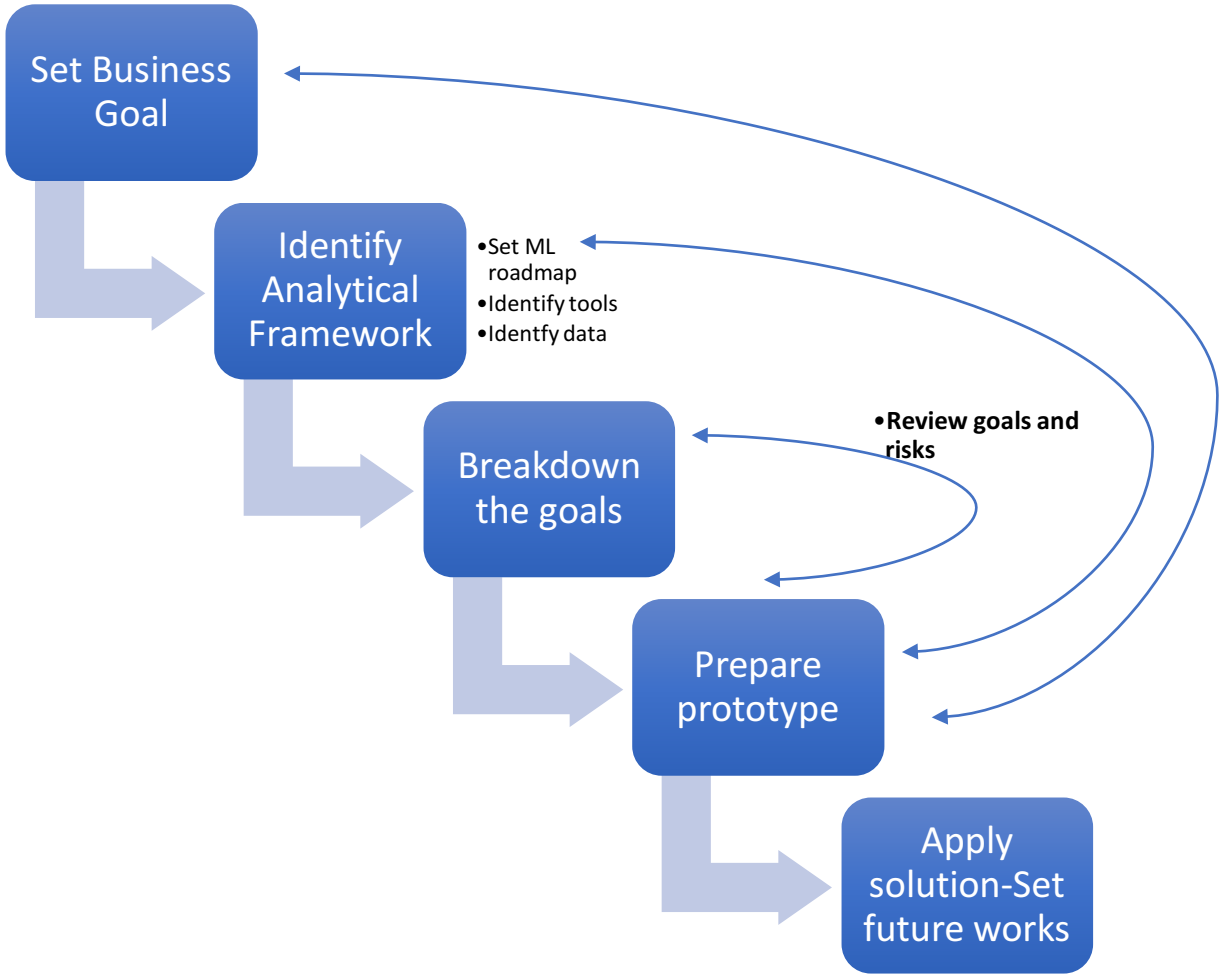


Figure 3.2 Research Methodology

Chapter 4

Developments and Findings

4.1 Introduction

This chapter will demonstrate the development and implementation of graph enhanced machine learning problem by using MovieLens Dataset. Details of the identification of the analytical framework and prototyping phase that is from the applied methodology will be explained in detail. For artefact design neo4j training and "Graph Algorithms Practical Examples in Apache Spark and Neo4j" (Needham and Hodler, 2019). was followed to understand the graph features and algorithms.

Identification of Analytical Framework:

4.2 Data Collection and Initial Analysis

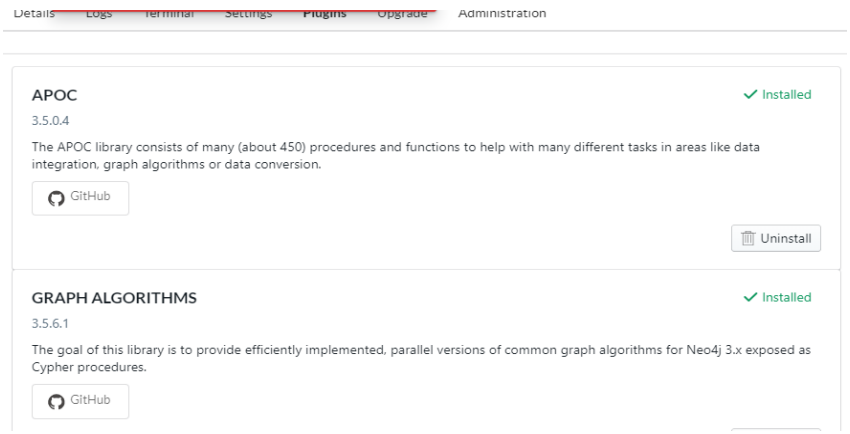
This thesis MoviLens data was used. This dataset has 100836 user ratings by 610 users across 9742 movies. There is no demographic information about the users. CSV files, Rating.csv and movie.csv, were used to create data model. Ratings file contains userId, movieId, rating, timestamp. Movie file contains movieId, title, genres. Release date of the movie represented in title column, and genres are pipe separated list. Grouplens (2019). For this research, during this phase it was decided to separate release date from the title and add as a property for the movie node and to separate genre and load as a node to the neo4j database.

With regarding the business goal, it was decided to solve this link prediction as a classification problem. Because random forest classifier good at the weak features, it was selected as a machine learning algorithm. This selection would be experimented on prototyping phase.

4.3 Tool Selection and Environment Settings for Development

The first step, before loading the data, was setting up the environment which was required for the development process.

Neo4j: Graph database that allows to use graph features for machine learning process. Apoc and graph algorithms were needed to install our database environment in order to use of graph algorithms.



After preparing neo4j environment, required packages were loaded to python in order to load movie dataset and to do data analysis.

py2neo: Python library for neo4j that provides connection for neo4j and execute cypher queries on neo4j.

pandas: Python library for data wrangling outside of a database, data analysis tools on python.

Prototyping Phase:

4.4 Data Loading and Data Modelling

After development environments were prepared, small changes were done to structure data in order to use them as a property in database. Such as, release date was separated from the movie title and added as a different column. User node were created from the rating.csv data for graph model. Csv files were loaded from python environment. For graph model movie – user ratings

and movie- genre relationships were created. Result of the initial loaded data model shown in below:

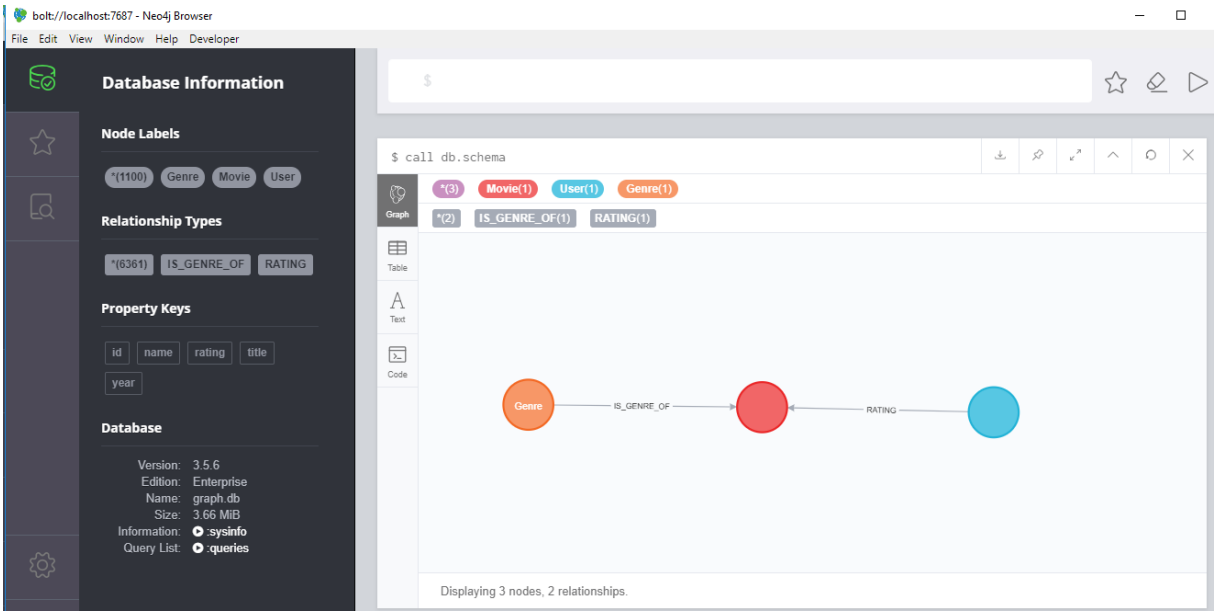


Figure 4.1 Initial model after data loading process

4.5 Data Analysis and Graph Feature Selection

Because the aim of this thesis is the prediction of missing links between users it is needed to set user-user relationship for every pair of users that have collaboration on movies by giving ratings. These relationships are important for machine learning model.

As source data had been cleaned and structured during the loading process, in this phase data was just split for training and testing purpose of machine learning algorithm. Release date was seen as a good splitting point for that purpose as shown in figure 4.2. Train and test set contains the relationship of users who rated the same movies in common. In this stage data was labelled for algorithm by finding the user node pairs, that had not any relationships, which are two and three hops away from each other. Consequently, users that had not any relationships added to the train and test dataset.

To find future user collaborations for movies by using ratings, user-user relationships would give an idea about which users rated the same movie in common and this relationship should carry the properties of how many times they rated the same movie. This would show the degree

of their collaboration. The following code was used for this aim to set user-user collaboration for the same movies:

MATCH (a1:User)-[:RATING]->(Movie)<-[:RATING]-(a2:User)

WITH a1, a2, Movie ORDER BY a1 with a1,a2,count(*)

AS collaborations MERGE (a1)-[co:COL_USER]-(a2) SET co.collaborations = collaborations

Forexample user_id 1 and user_id 80 has rated 5 movies in common shown in figure 4.2.

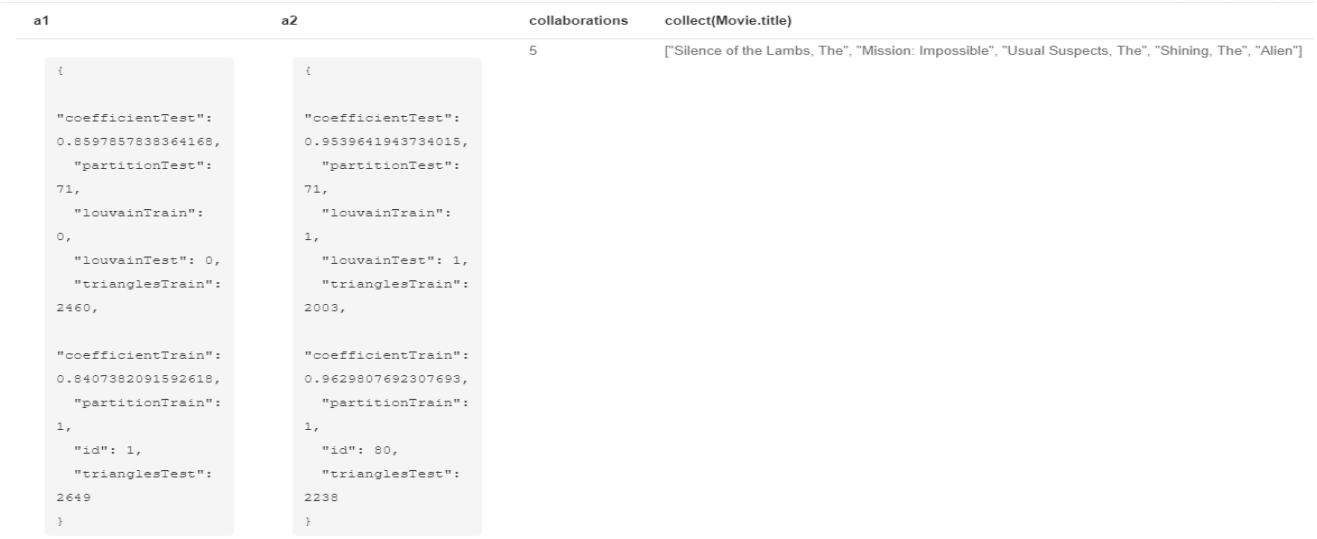


Figure 4.2 Users who watched same movies and how many times they rated them

By using this relationship test and train dataset also was prepared. Splitting movie data from the release date from 1995 gave acceptable test and train ratio as shown in figure 4.3.

MATCH (movie:Movie) RETURN movie.year < '1995' AS training, count(*) AS count

training	count
false	411
true	590

Figure 4.3 Determination of test and train data

After setting the new relationships for machine learning model, final graph model as shown in figure 4.4:

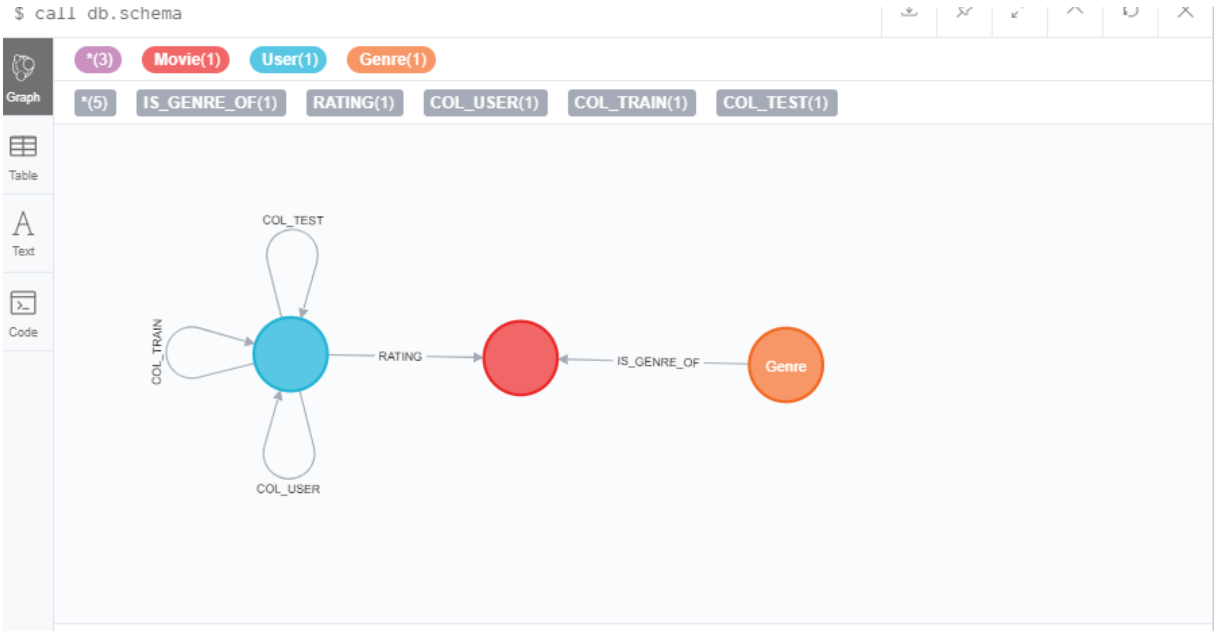


Figure 4.4 Final data model on graph for machine learning algorithm

Following code give how many pairs of nodes have relationship in test and data set:

"MATCH ()-[:COL_TRAIN]->()
RETURN count(*) AS count"

	count
0	2599

"MATCH ()-[:COL_TEST]->()
RETURN count(*) AS count"

	count
0	1846

Graphical representation of relationships in python shown in below the figure:

```
In [13]: rels_df.plot(kind='bar', x='relType', y='count', legend=None, title="Relationship Cardinalities")
         plt.xticks(rotation=45)
         plt.tight_layout()
         plt.show()
```

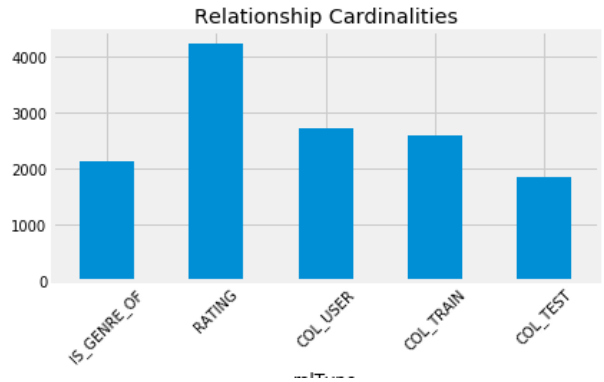


Figure 4.5 Node relationships

```
In [12]: result = {"relType": [], "count": []}
         for relationship_type in graph.run("CALL db.relationshipTypes()").to_series():
             query = f"MATCH ()-[:`{relationship_type}`]->() RETURN count(*) as count"
             count = graph.run(query).to_data_frame().iloc[0]['count']
             result["relType"].append(relationship_type)
             result["count"].append(count)
         rels_df = pd.DataFrame(data=result)
         rels_df.sort_values("count")
```

Out[12]:

	relType	count
4	COL_TEST	1846
0	IS_GENRE_OF	2118
3	COL_TRAIN	2599
2	COL_USER	2728
1	RATING	4243

Figure 4.6 Count of each relationships

Train and test data were built by using both nodes have relationships. It is needed to find and add node pairs to train end test samples that do not have relationships 2-3 hops away. In order to use binary classifier, data had to be labelled by whether users have a relationship or not. The results shown below illustrate relationships between node pairs that 0 indicates node pairs have not relations and 1 indicates node pairs have relation.

Out[34]:

	node1	node2
label		
0	1202	1202
1	2599	2599

Figure 4.7 Result of labelled data

4.6 Algorithm and Graph Feature Selection

For this prediction problem, some assumptions were made to find suitable graph features such as, users in the same clusters and users with more relationships would be increase the probability of possible links.

In this classification problem which was whether users will have a link or not, random forest classifier was selected as the machine learning algorithm. A random forest is ensemble methods that are collection of more than one decision trees. Each of the tree is trained on a randomly

selected subset of the attributes. In a random forest, each of the tree looks some of the attributes which is the distinction of random forest when compared with single decision tree. Each of the trees act as a weak classifier and merge together to predict target variable. Therefore, model sure that the weak features also is used. Graph features were added to classifier and thus improvement of the results was observed.

From the beginning of the implementation common neighbors (cn), preferential attachment (pa), and total neighbors (tn) were selected as graph features. After that, triangles and clustering coefficients and louvain features were added to the model and subsequently the model was evaluated. The evaluation was done by accuracy, precision and recall values.

The final step before running the algorithms was adding the graph features on test and train set. After adding the graph features final dataset for test data shown in figure 4.8.

```
In [44]: test_df.head()
```

Out[44]:

	label	node1	node2	cn	pa	tn
0	0	9	9	0.000	6241.000	79.000
1	0	9	40	43.000	3476.000	80.000
2	0	9	15	34.000	2765.000	80.000
3	0	9	67	61.000	4898.000	80.000
4	0	9	104	75.000	6004.000	80.000

Figure 4.8 Final dataset for test set

Because of its efficiency first of all common neighbor (cn) was added to the model. Common neighbor would show results illustrating that more connected nodes have increased possibilities for the future of new connections. The result of accuracy %64 were assumed as reasonable because data was sampled.

“It should be noted that, resulting score using CN is not normalized, and only shows the relative similarity of different node pairs by considering shared nodes between them.” (Mutlu and Oghaz, 2019)

```
In [45]: columns = ["cn"]

         X = training_df[columns]
         y = training_df["label"]
         classifier.fit(X, y)

         predictions = classifier.predict(test_df[columns])
         y_test = test_df["label"]

         display("Accuracy", accuracy_score(y_test, predictions))
         display("Precision", precision_score(y_test, predictions))
         display("Recall", recall_score(y_test, predictions))

         sorted(list(zip(columns, classifier.feature_importances_)), key = lambda x: x[1]*-1)

         'Accuracy'

         0.6424980153479757

         'Precision'

         0.5829701642641636

         'Recall'

         0.9420368364030336
```

Figure 4.9 Effect of common neighbors

Then, preferential attachment and total neighbors were added to the model. Higher preferential attachment score would give high degree for the future link. As shown in figure 4.10 model performance in recall was increased by adding preferential attachment and total neighbors. Similarity based features has significant effect on model results.

```
In [46]: columns = ["cn", "pa", "tn"]

         X = training_df[columns]
         y = training_df["label"]
         classifier.fit(X, y)

         predictions = classifier.predict(test_df[columns])
         y_test = test_df["label"]

         display("Accuracy", accuracy_score(y_test, predictions))
         display("Precision", precision_score(y_test, predictions))
         display("Recall", recall_score(y_test, predictions))

         sorted(list(zip(columns, classifier.feature_importances_)), key = lambda x: x[1]*-1)

         'Accuracy'

         0.6477904207462292

         'Precision'

         0.5843985578498853

         'Recall'

         0.9658721560130011
```

```
Out[46]: [('cn', 0.4208491606077223),
          ('pa', 0.3539778420232224),
          ('tn', 0.22517299736905536)]
```

Figure 4.10 Effect of common neighbors, preferential attachment and total neighbors and importance of features

It was seen that Similarity based features was increased model accuracy. Community detection algorithms were also examined on the model. These results showed that the Triangles and The Clustering Coefficient also had a positive effect on model results as illustrated figure 4.11.

```
In [72]: columns = [
             "cn", "pa", "tn", # graph features
             "minTriangles", "maxTriangles", "minCoefficient", "maxCoefficient" # triangle features
         ]

         X = training_df[columns]
         y = training_df["label"]
         classifier.fit(X, y)

         predictions = classifier.predict(test_df[columns])
         y_test = test_df["label"]

         display("Accuracy", accuracy_score(y_test, predictions))
         display("Precision", precision_score(y_test, predictions))
         display("Recall", recall_score(y_test, predictions))

         sorted(list(zip(columns, classifier.feature_importances_)), key = lambda x: x[1]*-1)

         'Accuracy'

         0.6371562709590879

         'Precision'

         0.5872274143302181

         'Recall'

         0.9856209150326798

Out[72]: [('pa', 0.22439769166680149),
          ('tn', 0.21514698234385896),
          ('cn', 0.16697214106164945),
          ('minTriangles', 0.1367787165456515),
          ('maxTriangles', 0.09305560301577712),
          ('maxCoefficient', 0.0897507897225371),
          ('minCoefficient', 0.0738980756437244)]
```

Figure 4.11 Effect of Triangles and Clustering Coefficient features and importance of all features

After that, triangle and clustering coefficient, louvain and label propagation which are community detection measures were added to the model. However, not significant change was observed on the results as shown in figure 4.12.

```
In [36]: columns = [
             "cn", "pa", "tn", # graph features
             "minTriangles", "maxTriangles", "minCoefficient", "maxCoefficient", # triangle features
             "sp", "sl" # community features
         ]

         X = training_df[columns]
         y = training_df["label"]
         classifier.fit(X, y)

         predictions = classifier.predict(test_df[columns])
         y_test = test_df["label"]

         display("Accuracy", accuracy_score(y_test, predictions))
         display("Precision", precision_score(y_test, predictions))
         display("Recall", recall_score(y_test, predictions))

         sorted(list(zip(columns, classifier.feature_importances_)), key = lambda x: x[1]*-1)
```

```
'Accuracy'

0.6277665995975855

'Precision'

0.5806451612903226

'Recall'

0.9882352941176471
```

```
Out[36]: [('pa', 0.23852833619351427),
          ('cn', 0.22227333789879708),
          ('tn', 0.17236806999215198),
          ('maxTriangles', 0.0846690872994002),
          ('minCoefficient', 0.07875106085414404),
          ('minTriangles', 0.07108819030678573),
          ('maxCoefficient', 0.068785016210453),
          ('sl', 0.06353690124475363),
          ('sp', 0.0)]
```

Figure 4.12 Effect of Louvain and Label Propagation features and importance of all features

Chapter 5

5.1 Evaluation:

Feature selection is one of the most important point for machine learning projects. Result of this thesis shows that graph based feature selection for machine learning problem has given reasonable outcomes. In database graph algorithms, on neo4j provide easy deployment during the implementation. Graph algorithms measures was added as input feature for machine learning model. This thesis proposes the effect of the graph features on machine learning outcomes by choosing link prediction problem. The accuracy, precision and recall was used as a performance metric for evaluation of the model. Graph features which were used in this research hypothesise that the closeness and connectivity would affect the creation of new possible links. In order to observe improvements on the model firstly, basic similarity graph features were extracted from common neighbors. The basic idea is that, two users who have common connections may connect in the future. By using these results as a base, preferential attachment and total neighbors added to the model. These graph measures result in the calculation of closeness of the nodes by using their neighbors. Results show that adding graph algorithm's based features has improved the model outcomes especially recall measurements. Between similarity based graph features, the common neighbors feature is the most influential on the prediction problem.

As a second step the effect of the community detection algorithms was examined to calculate groups and their scores. Triangle and clustering coefficient features have improved the recall score up to %98 whereas the accuracy slightly decreased from %64 to %63. In addition to the similarity based features, minimum triangle coefficient features become important on the model after adding the community features. Adding more community detection algorithm's, louvain and label propagation, features have not improved the scores significantly. Results has also showed that label propagation features do not have effect on this problem. Importance of the features after adding community detection algorithms are listed in figure 4.10.

5.2 Limitations

Firstly, graph enhanced machine learning is a new research are therefore, finding information and related works was relatively difficult.

Secondly, both neo4j and python require memory for computations. 6 GB memory was not enough for developments. As a result of this restriction, developments were done by sample size.

Chapter 6

Conclusion and Future Works:

Graph technologies provide an advantage on connected data. The purpose of this study is to use this advantage to explore the impact of the graph features on recall measure on link prediction problem. Results obtained from the movilens dataset, which shows that node similarities and closeness graph measures are influential on this supervised learning model. It is observed that graph derived features improve the classification recall measure. Additionally, in database algorithms on neo4j makes feature analysis easier.

During the implementation process, the methodology has been studied and the steps that have been applied for this research was proposed. This methodology was followed during the implementation of different graph algorithms to find the most influential graph feature. This study on graph technologies demonstrates that graph measures improve recall measures on the link prediction problem. This proposed methodology can be used for different analytical studies and graph data models.

Analysis on connected data and graph features will support the decision-making process and give consumable and insightful information for further artificial intelligence studies. For future studies, more nodes and relationships could be added to extend the current movie graph model. Other graph algorithms and derived features could be experimented with to increase accuracy. Furthermore, extended graph model could be converted to a vector as a graph embedding and will be able to be used for graph neural networks.

References:

Dastyar, B., Kazemnejad, H., Sereshgi, Alireza A.; Jabalameli, Amin M. (2017)' Using Data Mining Techniques to Develop Knowledge Management in Organizations: A Review 'Journal of Engineering, Project & Production Management., 7(2) pp. 80-89., EBSCOhost [Online]. (Accessed: 2 March 2019)

De Veaux, Richard D.; Hoerl, Roger W.; Snee, Ronald D. (2016) 'Big data and the missing links', Statistical Analysis & Data Mining, 9(6), p411-416. 13p., EBSCOhost [Online]. (Accessed: 2 March 2019)

F. Maxwell Harper and Joseph A. Konstan. 2015. The MovieLens Datasets: History and Context. ACM Transactions on Interactive Intelligent Systems (TiiS) 5, 4: 19:1–19:19. Available at: https://doi.org/10.1145/2827872 (Accessed: 20 August 2019)

'Newsroom', 'Gartner Identifies Top 10 Data and Analytics Technology Trends for 2019' [Online] Available at: https://www.gartner.com/en/newsroom/press-releases/2019-02-18-gartner-identifies-top-10-data-and-analytics-technolo (Accessed 10 August 2019)

Grouplens (2019)'Movilens', grouplens, [Online] Available at: https://grouplens.org/datasets/movielens/ (Accessed: 20 August 2019)

Gupta, S., Pandey, S. and Shukla, K.K. (2015) 'Comparison Analysis of Link Prediction Algorithms in Social Network', International Journal of Computer Applications, 111(16), [Online]. Available at https://pdfs.semanticscholar.org/bb5e/9530cd210474a2b7ac5b263c8eea888a740e.pdf (Accessed: 25 May 2010).

Koh, H.C., Tan G. (2011) "Data mining applications in healthcare", Journal of healthcare information management [Online]. Available at: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.92.3184&rep=rep1&type=pdf (Accessed: 28 Feb 2019)

Mariscal, G., Marbán, Ó., Fernández, C. (2010) "A survey of data mining and knowledge discovery process models and methodologies", The Knowledge Engineering Review, 25(2), pp. 137–166. EBSCOhost [Online] (Accessed: 28 Feb 2019)

MARTÍNEZ, V., BERZAL, F. and CUBERO, J.-C. (2017) 'A Survey of Link Prediction in Complex Networks', ACM Computing Surveys, 49(4), p. 69:1-69:33. Available at: http://search.ebscohost.com/login.aspx?direct=true&AuthType=ip,shib,cookie,url&db=iih&AN=127315662&site=eds-live (Accessed: 18 August 2019)

Moro,S. and Laureano, R., M. S., Cortez P. (2016) "USING DATA MINING FOR BANK DIRECT MARKETING: AN APPLICATION OF THE CRISP-DM METHODOLOGY", [Online] Available at: https://repositorium.sdum.uminho.pt/bitstream/1822/14838/1/MoroCortezLaureano_DMApproach4DirectMKT.pdf (Accessed: 28 Feb 2019)

Mutlu, C.E. and Oghaz, A.T. (2019) "Review on Graph Feature Learning and Feature Extraction Techniques for Link Prediction", arXiv:1901.03425 [cs.SI]. Available at: https://arxiv.org/abs/1901.03425 (Accessed: 16 August 2019).

Najera, J. (2010) 'Graph Theory — History & Overview', Towards Data Science. Available at: https://towardsdatascience.com/graph-theory-history-overview-f89a3efc0478 (Accessed: 24 May 2019)

Needham, M. and Hodler, E.A. (2019) “Graph Algorithms Practical Examples in Apache Spark and Neo4j”. 1th edn. O’Reilly. Neo4j [Online]. Available at: https://neo4j.com/lp/book-graph-algorithms/?utm_source=google&utm_medium=ppc&utm_campaign=*UK%20-%20Search%20-%20Graph%20Algorithms&utm_adgroup=*UK%20-%20Search%20-%20Graph%20Algorithms%20-%20Graph%20Algorithm&utm_term=graph%20algorithms&gclid=CjwKCAjwqNnqBRATEiwAkHm2BHXF1HEV9VYn5FUKpqHCyEfecUxTDLOr5erVARrzz1XTeS6X03DNvRoCRREQAvD_BwE (Accessed: 03 Jul 2019)

‘The Neo4j Graph Algorithms User Guide v3.5’ (no date). Available at: https://neo4j.com/docs/graph-algorithms/current/ (Accessed: 10 August 2019)

PAWAR, Asha M. (2016) “Big Data Mining: Challenges, Technologies, Tools and Applications”, Database Systems Journal. 2016, 7(2), pp. 28-33. EBSCOhost [Online] (Accessed: 28 Feb 2019)

Battaglia, P. W., Hamrick, J. B., Bapst, V., Sanchez-Gonzalez, A., Zambaldi V., Malinowski, M., Tacchetti, A., Raposo, D., Santoro, A., Faulkner, R., Gulcehre, C., Song, F., Ballard, A., Gilmer, J., Dahl, G., Vaswani, A., Allen, K., Nash, C., Langston, V., Dyer, C., Heess, N., Wierstra, D., Kohli, P., Botvinick, M., Vinyals, O., Li, Y. and Pascanu, R. (2018)“Relational inductive biases, deep learning, and graph networks,” , arXiv:1806.01261 [cs.LG]. Available at: https://arxiv.org/abs/1806.01261 (Accessed: 5 August 2019)

Robinson, I., Webber, J. and Eifrem, E. (2015) Graph Databases. 2th edn. 1005 Gravenstein Highway North, Sebastopol, CA 95472 O’Reilly Media, Inc. graphdatabases [Online]. Available at: https://graphdatabases.com (Accessed: 24 May 2019).

Shearer, C. (2000) 'Journal of Data Warehousing' pp. 13-18.

‘What is a Graph Database?’ (no date). Available at: https://neo4j.com/developer/graph-database/ (Accessed: 3 Jun 2019)

Wilson, J.R. (2010) Introduction to Graph Theory. Edinburgh Gate, Harlow: Pearson Education Limited.

Yang, Y., Lichtenwalter, R. and Chawla, N. (2015) ‘Evaluating link prediction methods’, Knowledge & Information Systems, 45(3), pp. 751–782. doi: 10.1007/s10115-014-0789-0.

Appendix:

This document will show the contents of the Artifacts and the necessary steps to implement the python code for dissertation project titled “Improvement of recall measure by deriving graph features for link prediction on machine learning algorithms”.

Contents of the Artifacts

1. **Datasets:**

Movie.csv
Ratings.csv

2. **Python Codes and Results:**

- **MovieDataLoadNEO4j-FINAL:** Data loading codes to neo4j
- **02_EDA-Movie:** Explatory data analysis on neo4j with python
- **04_Prediction_2-User80-FINAL:** Development and results for 80 users and their relationships
- **04_Prediction_2-User100-movie1000Final:** Development and results for 100 users and their relationships
- **04_Prediction_2-User200-movie-1000-FINAL:** Development and results for 200 users and their relationships

3. **Cypher queries:** For Cosine similarity and relationships