Recommendation System by Link Prediction Approach on Transactional Data

by

Emir Alaattin Yılmaz

Submitted to the Graduate School of Engineering and Natural Sciences in partial fulfilment of the requirements for the degree of Master of Science

Sabancı University

July 2021

RECOMMENDATION SYSTEM BY LINK PREDICTION APPROACH ON TRANSACTIONAL DATA

Approved by:

Date of Approval: 02/07/2021

Recommendation System by Link Prediction Approach on Transactional Data

Emir Alaattin Yılmaz

Computer Science and Engineering Master's Thesis, July 2021

Thesis Supervisor: Prof. Dr. Selim Saffet Balcısoy

Keywords: recommendation system, link prediction, transaction data, merchant prediction

ABSTRACT

With the continuous digitalization of the world, massive amounts of data are produced every second. Therefore, recommending relevant items to users has become a more important task in many systems. For this purpose, transaction data sets can be exploited in recommendation systems to understand underlying user interests. Commonly used recommendation systems adopt collaborative filtering based approaches that utilize a user-item matrix based on users' past activity. However, these methods may suffer from sparsity and scalability issues. In this thesis, a link prediction based recommendation system combining graph representation learning algorithms and gradient boosting classifiers for transaction data sets is proposed as a scalable solution. Proposed system generates a network where nodes correspond to users and items, and links represent the interactions between them. A use case scenario is examined on a credit card transaction data set as a merchant prediction task which is predicting the merchants where users can make purchases in the next month, in a link prediction context. Performances of common network embedding extraction techniques and classifier models are evaluated by conducted experiments, and based on these evaluations; the proposed system is constituted. A matrix factorization based alternative scalable recommendation method is compared with the proposed model. Proposed method has shown a superior performance than alternative method in terms of receiver operating characteristic curves, area under the curve, and mean average precision metrics.

İşlem Verilerinde Bağlantı Tahmini Yaklaşımı ile Öneri Sistemi

Emir Alaattin Yılmaz

Bilgisayar Bilimi ve Mühendisliği Yüksek Lisans Tezi, Temmuz 2021

Tez Danışmanı: Prof. Dr. Selim Saffet Balcısoy

Anahtar Kelimeler: öneri sistemi, bağlantı tahmini, işlem verisi, işyeri tahmini

ÖZET

Dünyanın süregelen dijitalleşmesi ile her saniye çok büyük miktarlarda veri üretilmektedir. Bu nedenle, kullanıcılara ilgili öğeleri önermek, birçok sistemde daha da önemli bir görev haline gelmiştir. Bu amaçla, öneri sistemlerinde altta yatan kullanıcı ilgilerini anlamak için işlem veri setlerinden yararlanılabilir. Yaygın olarak kullanılan öneri sistemleri, kullanıcıların geçmiş etkinliklerine dayalı bir kullanıcı öğe matrisinden faydalanan işbirlikçi filtrelemeye dayalı yaklaşımları benimsemektedir. Ancak bu yöntemler seyreklik ve ölçeklenebilirlik sorunları yaşayabilirler. Bu tezde, işlem veri setleri için grafik temsili öğrenme algoritmalarını ve gradyan artırıcı sınıflandırıcıları birleştiren, bağlantı tahmini tabanlı bir öneri sistemi, ölçeklenebilir bir çözüm olarak önerilmiştir. Önerilen sistem, düğümlerin kullanıcılara ve öğelere karşılık geldiği, bağlantıların aralarındaki etkileşimleri temsil ettiği bir ağ oluşturur. Bir kredi kartı işlem veri seti üzerinde, kullanıcıların bir sonraki ay içerisinde alışveriş yapabilecekleri işyerlerini tahmin eden bir üye işyeri tahmin görevi, bir kullanım senaryosu olarak bağlantı tahmini bağlamında incelenmiştir. Yaygın olarak kullanılan ağ gömme çıkarımı teknikleri ve sınıflandırıcı modellerinin performansları, deneyler yapılarak değerlendirilmiş ve bu değerlendirmelere dayalı olarak önerilen sistem oluşturulmuştur. Önerilen model ile matris çarpanlarına ayırma tabanlı alternatif bir ölçeklenebilir öneri yöntemi karşılaştırılmıştır. Önerilen yöntem, alıcı çalışma karakteristik eğrileri, eğri altında kalan alan ve ortalama kesinlik değerlerinin ortalaması metrikleri açısından alternatif yönteme göre daha üstün bir performans göstermiştir.

ACKNOWLEDGEMENTS

First and foremost, I would like to express my sincere gratitude to my supervisor Professor Selim Saffet Balcısoy, for providing valuable mentoring and great support throughout my master's study. I am very grateful for his scientific guidance and encouragement of new ideas.

Also, I would like to thank Prof. Dr. Burçin Bozkaya for his invaluable feedback about my thesis study. I also would like to thank Prof. Dr. Yücel Saygın for his precious time to be present in my thesis jury.

I thank Kerim Gökarslan, Muratcan Serdar Canlı, Hamza Kandiş, Alphan Yalaz, Yağız Varol, Işın Mert Balcı, Berkant Deniz Aktaş, Hasan Alp Boz, Ahmet Ege Güldalı, Ömer Burak Aladağ, Cansu Ekiz, Ayşe Merve Soylu for their great friendship.

I also thank Anıl Özdemir, Furkan Coşkun, Ethem Tunal Hamzaoğlu, Pınar Ön for their support for my master's study and their great friendship.

And finally, I would like to thank my beloved family Gamze Yılmaz, Mehmet Yılmaz, and Mehmet Akif Yılmaz, for their constant support throughout my entire life.

"Science is the only true guide in life."
Mustafa Kemal ATATÜRK

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

ALS: Alternating Least Squares

ANN: Artificial Neural Networks

AUC: Area Under the Curve

CF: Collaborative Filtering

GBDT: Gradient Boosting Decision Tree

GT: Ground Truth

LGBM: Light Gradient Boosting Machine

MAP@K: Mean Average Precision at K

MCN: Merchant Customer Network

PCA: Principal Component Analysis

ROC: Receiver Operating Characteristic

SVD: Singular Value Decomposition

t-SNE: t-Distributed Stochastic Neighbor Embedding

UIN: User-Item Network

XGB: Extreme Gradient Boosting

Chapter 1

Introduction

Graphs are data structures that can powerfully abstract complex systems. Many real-world systems can be represented in a graph structure where individuals and relationships between them correspond to nodes and edges. These systems include social networks [3, 4], biological networks [5, 6], physical networks [7], citation networks [4], knowledge representations [8] and much more [9]. Due to its applicability in a wide range of domains, graph systems advancements can lead to great improvements in many diverse fields.

The systems that exploit network architectures can perform common tasks such as node classification, community detection, and link prediction. Node classification is to decide the class of a node in a graph; community detection is to find the clusters that contain similar nodes in a network, and link prediction is to discover hidden links based on observed links.

Link prediction has many diverse applications in real-world scenarios such as predicting the future friends in a social network, detecting future collaborations in an academic network [4], finding protein-protein interactions (PPI) in a biological network, and recommending items in recommendation systems. Due to its high practicality, research in this field can lead to many contributions in a wide range of systems. Among these applications, recommendation systems are the primary consideration of this thesis.

Recommendation systems offer items to users which they might be interested in. These items can be relevant links on a search engine, products on an e-commerce website, movies, and much more. Common recommendation systems use collaborative filtering based approaches that utilize a user-item matrix to make recommendations using users' past activity. However, these methods may have sparsity and scalability issues [10].

A link prediction approach for a recommendation system can operate in a small neighborhood which can be proposed as a scalable solution for the scalability issues in recommendation systems. In addition, transaction data sets consisting of records with item-user pairs, timestamps, and optional numerical quantities can be exploited to understand user interests and make recommendations. Based on these motivations, a link prediction based recommendation system for transaction data sets is proposed in this thesis.

In this thesis, a link prediction based recommendation system for transaction data sets is presented with the aim of predicting and recommending relevant items which the user can interact but not linked up with these items before.

Proposed method in this thesis adopts a binary classification approach to link prediction problem where the actual links and non-existent links are classified as "true" and "false", respectively. To perform this task, it includes preprocessing step, embedding model and classifier model. Transaction data is processed at preprocessing step to create a graph network where nodes are users and items; links are the associations between them. Embedding model extracts the network information and puts it in a low dimensional form (node embeddings) which can be used in machine learning tasks. Extracted node embeddings are put in a link embedding form by a binary operator. Then, embeddings of existent and non-existent links are used to train a classifier model and then this model is used to classify a link as "true" or "false" by thresholding. As a use case scenario, a credit card transaction data set is used to predict and recommend merchants which customers might make purchases in the next month based on their past transactions.

The contributions of this thesis are the following:

- Performances of embedding extraction methods node2vec [1], metapath2vec [11] and binary operators averaging, hadamard, l2 and classifier models XGBoost [12], LightGBM [13], Artificial Neural Networks are compared to find the best combination on the use case scenario. Based on the evaluations on area under the curve (AUC) and mean average precision at K (MAP@K) metrics, metapath2vec, averaging, LightGBM is found to be the best combination and presented as the proposed model.

- Performance of a collaborative filtering based alternative scalable recommendation method Alternating Least Squares [14] is compared with the proposed model. Proposed method has shown superior performance than alternative method in terms of receiver operating characteristic (ROC) curves, area under the curve (AUC) and mean average precision at K (MAP@K) scores.

In Chapter 2, a literature review is conducted on existing link prediction, graph representation learning, and recommendation systems. Chapter **??** gives the definitions and describes the research problem. Chapter 3 explains the proposed method in detail with theoretical background. Chapter 4 gives details about implementation process. Chapter 5 presents a use case scenario as merchant prediction on a transaction data set. In Chapter 6 experimental results are evaluated. Lastly, in Chapter 7 conclusion of research and future works that can expand the research is given.

Chapter 2

Related Works

In recent years, many methods have been proposed in link prediction, graph representation learning, classification algorithms and recommendation systems. Throughout this chapter, commonly used methods in these fields will be discussed to strengthen the core of this research.

Link Prediction Algorithms

There are several approaches on link prediction task. The most commonly used methods are similarity-based algorithms. Similarity-based algorithms assume that two similar nodes will interact with a high probability [15]. These methods assign a similarity score for pair of nodes and create an ordered list of links based on these scores. Defining the similarity has different types of interpretations. Some methods focus on commonality of nodes at feature level [16, 17]. Although this is a reasonable approach, node features may not always be present. Therefore, structural similarity methods are taken into consideration where network topology information such as node degree, neighborhood commonality and links between common neighborhood can be used. Similarity indices can be classified as local or global. Local similarity indices such as common neighbours [18] resembles to determining mutual friends of two people in a social network, Salton index [19] which is also known as cosine similarity, Sørensen Index [20] considers common neighbors and states nodes with lower degree have higher link probability, Jaccard Index [21] calculates the ratio of intersection over union and is introduced many years ago, Adamic/Adar index [22] which is calculated as summing log-degree centrality of two nodes are used frequently. Furthermore, global indices such as Katz centrality [23] that measures centrality of a node by considering the total number of walks between pair of nodes also have similarities between famous page-rank algorithm [24] and eigenvector centrality [25], local path [26] metric which considers local paths are also considered.

Another type of link prediction methods is maximum likelihood estimation-based algorithms. These methods optimize parameters that maximize the likelihood of observing network structure. These parameters can also be used to calculate the probability of non-observed links. However, they have some disadvantages such as high computational time complexity and giving sub-optimal results [27]. One of the maximum likelihood-based algorithms is Hierarchical structure model [28] uses dendrograms that create hierarchical organizations of networks which are used to infer link probabilities for missing links. Stochastic Block Models [29, 30] is a generative model which organizes nodes into blocks that has the same structural patterns of connections. It can capture modules, clusters, and communities [31]. This method assigns probabilities of link existence to pairs of nodes based on their group memberships by using Bayesian inference.

Learning-based models adopt a probabilistic approach that generates a joint probability distribution that abstracts the observed graph by optimizing an objective function including model parameters. Based on this distribution, link existence and non-existence probabilities can be calculated using conditional probability. Probabilistic relational models [32, 33] create a joint probability that abstracts attributes of relational data sets. This method includes three types of networks: skeleton as data graph, model graph, and ground as inference graph. This technique considers the type of the nodes, dependencies among item types, and probabilistic dependencies of variables. Inference graph can be built as Bayesian Networks, Markov Networks, or Dependency Networks. Stochastic relational models [34] considers the random nature of links by use of tensor factorization of multiple Gaussian Processes. Each process is associated with the type of links. By maximizing marginalized likelihood, the relational networks convey dependency information to Gaussian processes. This method presents a discriminative way for link prediction.

Related to learning-based models, graph neural networks (GNN) have gained attention in recent years for link prediction task. SEAL [35] is a graph neural networks based model to perform link prediction. In this method, GNN learns the general network information by extracting local enclosing sub-graphs and using appropriate heuristics for each related sub-graphs accordingly. Also, this method challenges the belief that nodes should be linked if they have common neighborhoods. Its prior work Weisfeiler-Lehman Neural Machine [36] adopts a similar approach. This method also extracts enclosing subgraphs and labels the nodes as their structural roles. Then, a neural network is trained by the adjacency matrix representation of these sub-graphs and used for link prediction.

Graph Representation Learning Algorithms

Graph representation learning algorithms are an essential part of link prediction methods. By these algorithms, graphs can be represented in a low-dimensional space by capturing network information. These low dimensional feature representations can be used in machine learning tasks such as anomaly detection, attribute prediction, clustering, and link prediction [37]. The classical methods of network feature extraction rely on network properties such as node degrees [38] which are label-independent features that can be a powerful signal while working on sparsely labeled data. Similarly, node-based features combined with egonet-based features can be used to capture behavioral information [39]. However, these methods conduct a hand-crafted feature extraction process which is not practical in many cases. Therefore, automation of this process can be a better approach to extract network representations.

Unsupervised representation learning algorithms do not involve hand-crafted feature extraction and make the whole process automated. Mainstream unsupervised methods use different types of matrix representations of graphs such as Laplacian and adjacency forms. Laplacian eigenmaps is a matrix factorization method [40] assumes that data lies in a low-dimensional manifold from a high-dimensional space. In principle, all of these linear algebraic methods can be interpreted as dimensionality reduction techniques. Dimensionality reduction techniques can be non-linear or linear. For linear algorithms, Principal Component Analysis (PCA) [41] which is the most common dimensionality technique that relies on the idea of generating uncorrelated variables that maximize the variance. Using these principle components corresponding to eigenvectors can represent the data in a low dimensional space. Similarly, Linear Discriminant Analysis (LDA) is another technique that tries to find a linear combination of features to maximize class separation that is different than PCA method. Furthermore, non-linear algorithms such as ISOMAP [42] tries to make nodes that are close to each other also close in terms of their geodesic distance in lower-dimensional space. Another non-linear technique t-SNE [43] minimizes the difference between conditional probabilities using Kullback-Leibler divergence that also minimizes the distance in low dimensional space. Although these dimensionality reduction methods are powerful, their computational complexity is high. Because eigendecomposition of a matrix is an expensive operation that is unfeasible to use in large networks. Another drawback of these methods is the lack of generalizability due to their strong assumptions on network structures such as homophily. Although homophily is effective for clustering, it is not applicable to all network structures with various patterns.

Recent advancements in natural language processing (NLP) techniques have opened a new path for representation learning. One of the noteworthy NLP methods is Word2vec [44] which creates vector representations of words from a large corpus. These representations or word embeddings are similar to each other for the words in the same or similar context. Skip-gram architecture in this technique uses the current word to predict the nearby words in a window of context words. Some graph representation learning algorithms have been developed based on this idea. These algorithms treat a graph as a large corpus and create random walks around each node representing sentences to learn latent representations. DeepWalk [45] is the first algorithm that applies this random walk idea on graphs to create low dimensional continuous node embeddings. LINE [46] is another method that also relies on concepts of NLP techniques. Instead of using random walk approach in DeepWalk that is analogical to depth-first search, LINE uses bread-first search method to create sentences and states that as a more reasonable approach. By doing this, it considers both the first order and the second-order proximity of nodes. Moreover, while DeepWalk can only be applied on unweighted edges, this is applicable to both weighted and unweighted edges. After these studies, node2vec [1] is developed by focusing on the sampling strategy of nodes which is the core component of these methods. node2vec combines both bread-first and depth-first search strategies by using parameters and performs biased random walks decided by these parameters at each step. By these parameters, the method can capture both local neighborhood and explore network structure and tries to maximize likelihood of network neighborhood. Due to its high flexible structure and generalizing power, it can capture network information better than DeepWalk (can also be interpreted as uniform random walks version of node2vec) and LINE.

Despite having convincing performances of graph feature extraction methods mentioned in both homogeneous and heterogeneous graphs, these methods have been developed considering that they will be used mostly in homogeneous graphs. Therefore, some other techniques have been developed, especially for heterogeneous graphs, to create better latent node representations. metapath2vec [11] have been developed for heterogeneous graphs, which takes types of the nodes into account while creating random walks. Random walks are generated based on meta-path schemes to create heterogeneous neighborhoods for different types of nodes. The goal of this method is to maximize the likelihood of heterogeneous network neighborhood.

Abovementioned graph representation learning algorithms can only generate representations for a single fixed graph which can not be generalized efficiently for unseen nodes due to their transductive nature. GraphSAGE [47] is an inductive method that uses node attributes to create node representations for unseen data.

Instead of separate node embeddings, it learns a function by creating a local neighborhood and aggregating feature information from the neighborhood. By use of this learned aggregation function, it can generate node representations for previously unseen nodes.

Boosting Algorithms

Boosting is a powerful technique introduced by Kearns and Valiant [48, 49] that adopts the approach of building a strong learner model based on the weak learners whose predictions are slightly better than random. A week classifier is trained, then another weak classifier focuses on the wrongly predicted data parts. Other week classifiers keep up this process in an iterative manner. As a result, a strong classifier is created. Adaboost [50] is the first algorithm using this boosting idea. Gradient boosting [51] is a generalized version of boosting algorithms by using gradient descent for optimization. Residuals are used in gradient boosting algorithms after each round as input to the next weak learner. Eventually, they are combined with a loss function to get an overall loss. The goal is to create a model that minimizes this overall loss. XGBoost [12] is a gradient boosting algorithm that introduces a regularization term in objective function to reduce overfitting. Moreover, training time is reduced by its parallelized and distributed architecture. Alongside these benefits, increase in data sizes can cause some scalability issues in gradient boosting algorithms. Feature selection and decreasing data size can be a solution to these problems. As another gradient boosting algorithm, LightGBM [13] developed by Microsoft introduces Gradient-based One-Side Sampling in gradient boosting. The data points which cause larger gradient changes are more preserved in this sampling strategy. Exclusive Feature Bundling technique groups the mutually exclusive features as bundles and treats them as a single feature to reduce the number of features.

Over the past decades, many different types of recommendation systems have been developed. They can be classified into three categories as content-based, collaborative filtering, and hybrid methods. In content-based methods [52] user descriptions based on their previous activity is used . User profiles are created from items in which the user shows an interest. Recommendation is achieved by calculating the similarity between item description and user profile. One of the limitations of content-based methods is the requirement of detailed user profiles, which is not always found. Item descriptions are in the form of textual information using TF-IDF [53, 54] and user profiles are encoded in the weighted combinations of these item description vectors [55]. Therefore, it provides an advantage to create a relation for previously unseen items.

Recommendation System Algorithms

Collaborative filtering is the most common recommendation system method which has various types of implementations. These methods can be divided into memory-based and latent factor models. Memory-based type also has sub-branches as item-based and user-based approaches. Item-based [56] approach predicts the rating of an item given from a particular user, based on the calculation of the ratings on similar items which had been rated by this user. However, in user-based [57] approach, this calculation is performed by considering other similar users' ratings for this item. Although memory-based techniques are easy to implement and have high interpretability, the recommendation matrix can be very sparse with many missing values, which is highly observed in real-life scenarios. Sparsity can decrease the performance and create scalability issues [58, 59]. On the contrary, latent factor models handles this sparsity problem by matrix factorization [60, 61, 62, 63] techniques. This method can be considered as a dimensionality reduction which creates latent factor representations approximating the recommendation matrix. This is usually achieved by Singular Value Decomposition (SVD) [64] which decomposes a sparse matrix into separate dense matrices. Some matrix factorization techniques can include parameter estimation steps [63] and use stochastic gradient descent algorithm for optimization [65]. However, in large data, scalability issues may arise due to stochastic gradient descent optimization. Because of that reason, a different optimization technique is used, which is Alternating Least Squares (ALS) [66]. In this optimization method, user interaction and item interaction matrices are fixed and derivatives are taken alternatively, making cost function quadratic, which can be optimized by alternating-least-squares method iteratively. Moreover, applications of the mentioned methods can vary according to the characteristics of data sets. Explicit data set has rating information from the users, but implicit data sets do not. A study [14] utilizes this Alternating Least Squares (ALS) approach and adapts it to explicit feedback data sets that give convincing performances.

Link Prediction as Recommendation System

Some studies use graph structures and link prediction methods as recommendation systems. A link prediction approach to collaborative filtering [10] uses some common similarity measures mentioned above. A resource allocation idea based approach [67] exploits bipartite graph structures and assigns weights to links by use of matrix projection of node degree information to provide personal recommendations. Another work [68] proposes a random-walk based recommendation framework that leverages node attributes into links as local, global or mixed weights, includes attribute ranking that simulates a friend hunting behavior. A link prediction based

recommendation approach [69] creates a graph where nodes are users and items, links are the associations between them assign weights to links as complex numbers to assess user similarity and user-item interest information. Another study about friend recommendation in social networks [70] extracts social patterns from different networks and transfers this information as attribute correlation and social correlation into a Markov Random Field while link recommending. Most of the link prediction based recommendation system methods [71, 59] are various adaptations of existing link prediction algorithms discussed above.

Chapter 3

Architecture

In this thesis, the research task is formulated as a link prediction problem. In link prediction context, a graph is generated from the transactions where nodes correspond to users and items, and links represent interaction occurrence between these users and items. Given this network generated from previous transactions, the goal here is to discover the new users-items links to be formed in a future time. While predicting these links, it can also be used as an item recommendation system.

Due to prediction characteristics of this task, a machine learning approach is adopted within the link prediction framework. To be more precise, this formulated problem is tackled as a binary classification task where actual links between the pair of nodes in the network are classified as "positive" and the links that do not exist classified as "negative".

3.1 Link Prediction Problem

Consider a graph $G = (E, V)$, where V is the set of vertices/nodes and E is the set of edges/links. Let E represent the "true" link set in the network and consider the case when only some parts of the links are given. These given links are the "observed" links and denoted as E_O where $E_O \subseteq E$. The task is to discover the hidden (unobserved) link set E_H where $E_H \cup E_O = E, E_H \cap E_O = \emptyset$.

For the temporal formulation of the link prediction problem, let E_t be the set of true links in a network observed at time t. The goal is to predict E_{t+1} which is the set of true links at time $t+1$.

Since the problem investigates link occurrences between pair of nodes, it can also be formulated as a binary classification task to examine if there is a link or

not. Links between nodes are considered as "true" links if there is a real connection among examined nodes, and "false" links if there is not an actual connection between these nodes in the network. A classifier is trained and tries to predict given test link is true or not. With regard to this formulation, a probability estimate of possible links can be associated with their probability of existence.

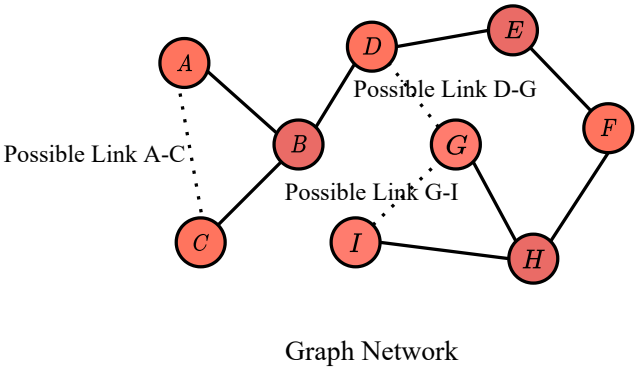

Figure 3.1 Visualization of link prediction problem

3.2 Research Problem Definition

Prediction and recommendation of the future items where the user can discover and interact in a future time by a link prediction approach is the main problem of this research. Task is to discover the new interactions that may occur in a future time given the past interactions by utilizing transactional data sets. By mean discover, it is actually to predict the most relevant items which the user can establish a connection but not interacted with these items before. These predictions can also be considered as recommendations which allows a link prediction framework to be used as a recommendation system.

3.3 Framework

In this thesis, the research task is formulated as a link prediction problem. In link prediction context, a graph is generated from the transactions where nodes correspond to users and items, and links represent interaction occurrences between these users and items. Given this network generated from previous transactions, the goal here is to discover the new users-items links to be formed in a future time. While predicting these links, it can also be used as an item recommendation system. Due to prediction characteristics of this task, a machine learning approach is adopted within the link prediction. To be more precise, the problem is tackled as a binary classification task where actual links between the pair of nodes in the network are classified as "positive" and the links that do not exist are classified as "negative".

At preprocessing step, transaction data is cleaned and processed to create a graph architecture where nodes correspond to users and items, and links represent the interactions. Embedding model captures network information and extracts low dimensional latent space representations of nodes (embeddings) that can be used as inputs to machine learning systems. A binary operator is applied to node embeddings to form link embeddings. Then, embeddings of existent and non-existent links train a classifier model which will be able to classify a link as "true" or "false" by applying certain thresholds. In this section, all stages will be explained in detail, including their theoretical backgrounds.

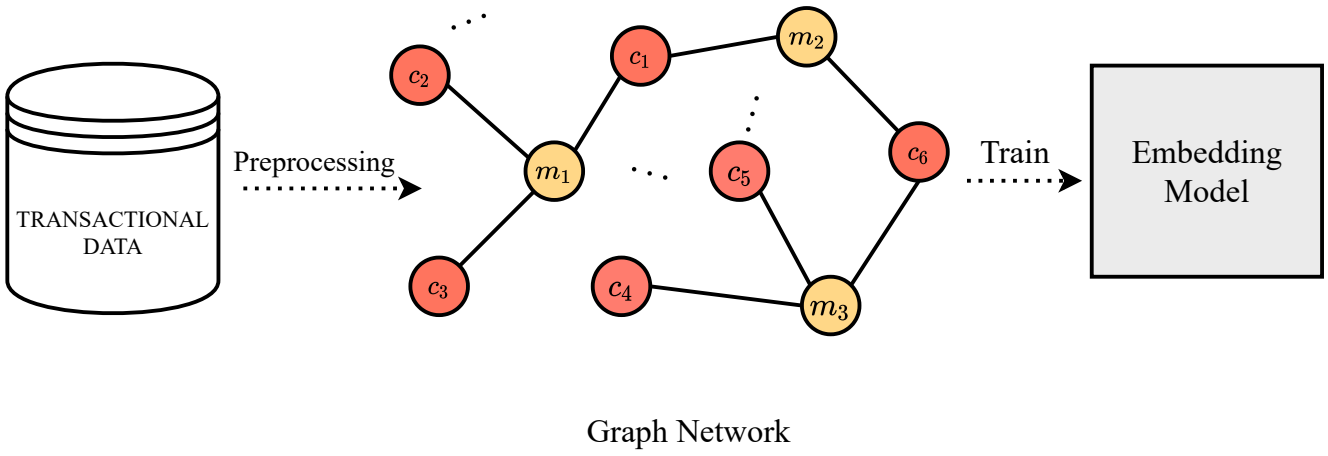

Figure 3.2 Illustration of preprocessing stage, graph network generation and embedding model training stages

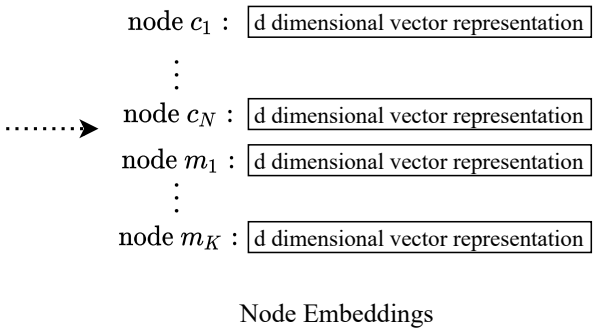

Figure 3.3 Node embeddings

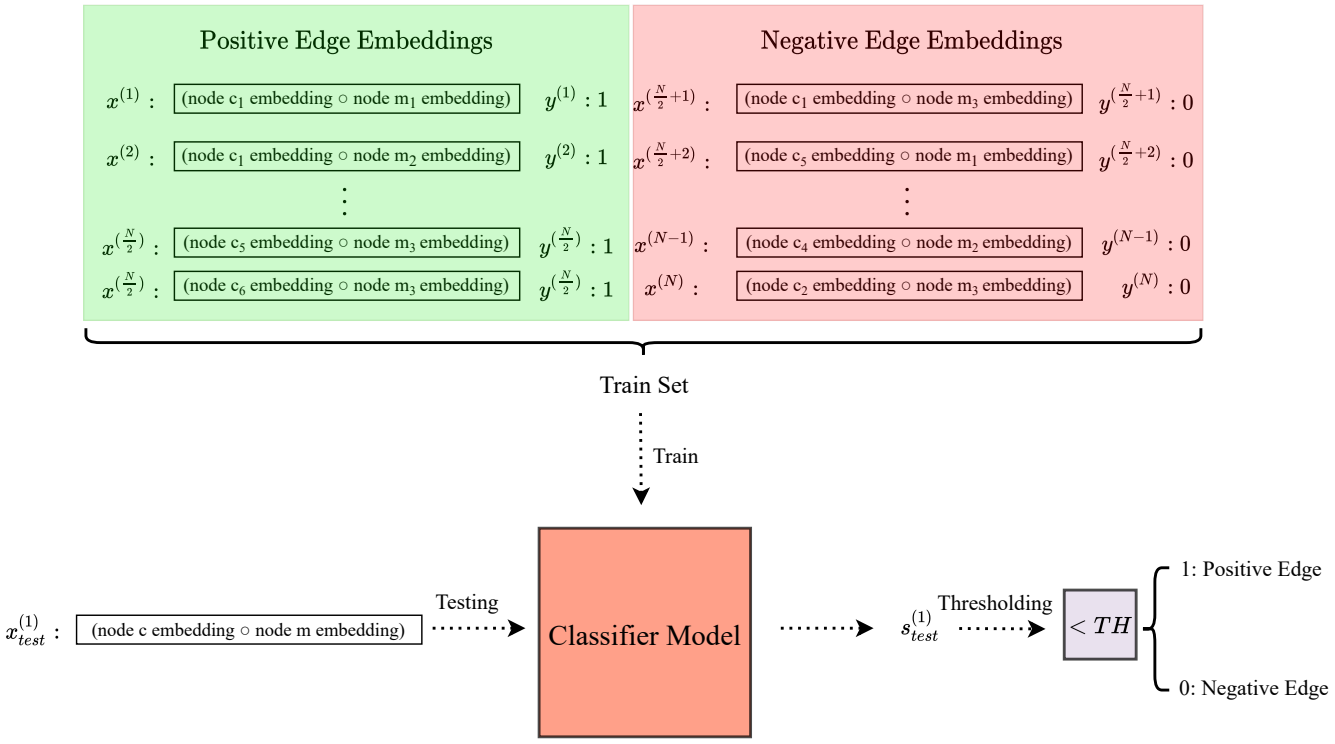

Figure 3.4 Illustration of train set generation by positive-negative edge embeddings and train of classifier

3.4 Preprocessing

User-Item Network (UIN) is a graph generated from the transaction history where nodes are users and items, and links are interaction occurrence of users with items.

There are two different types of nodes (users and items) in UIN. Therefore, this generated network is a heterogeneous graph. Moreover, since these links only exist between users and items, not between any user-user or item-item, this network also shows a bipartite network characteristic in its nature.

3.5 Node Embeddings

Graphs are architectures that have high abstractional power of many systems. However, they may have complex topologies which are hard to use in machine learning tasks. In order to benefit from their expressive power in machine learning tasks, there is a requirement to transform them into low-dimensional vectors. In this way, these d-dimensional vector representations can be used to generalize the network and can be given as input to machine learning models to perform prediction tasks. Also, in comparison with traditional graph analysis algorithms, which use inflexible and high costly hand-crafted features, graph representation learning algorithms have better generalization ability.

By representing the nodes in embedding space, similar nodes in the graph will also have similar embeddings. These methods can be considered as a sub-branch of word-embedding concept. In word-embedding methods, similar words also have similar embeddings. As a result, the words in the same context are also close in the embedding space.

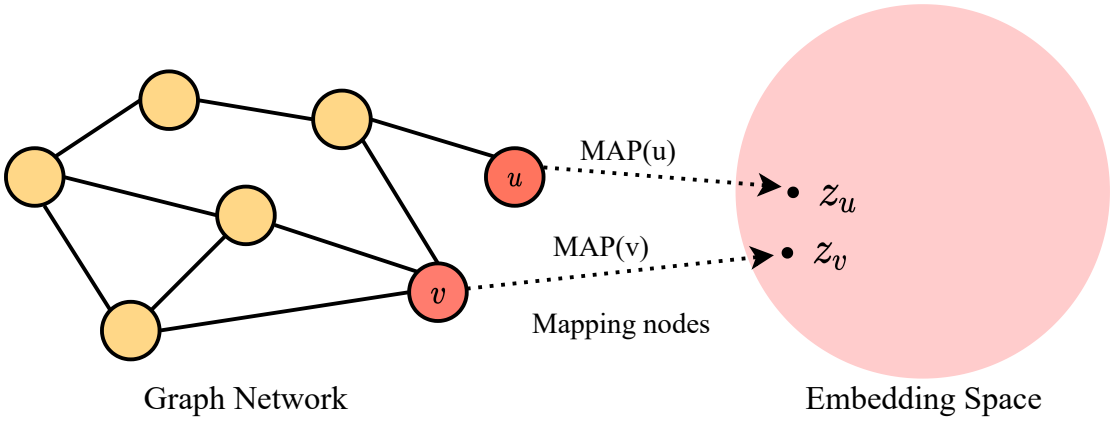

Figure 3.5 Representation of node u and node v in embedding space, z_u and z_v are embeddings of node u and v adopted from [2]

As can be seen in Figure 3.5, a mapping function encodes nodes into a d-dimensional embedding space. The goal is to make this mapping function approximate the network similarity of nodes to vector similarity of embeddings (dot product).

$$similarity(u, v) \approx z_v^T z_u$$

As mentioned earlier, positive links are the actual links that are present in the graph that also creates MCN, and negative links are the non-existing links. Nodes and corresponding positive links are used to train embedding model. By doing so, node embeddings are obtained after this process. These embeddings capture the network information which can be used in the machine learning tasks.

In this research, common node embedding extraction methods in the literature, namely node2vec and metapath2vec are used and examined in terms of their performances for the link prediction task in User-Item Network.

3.5.1 node2vec

Node2vec is the most common node representation method that is used to generate node embeddings from the networks. It optimizes a graph-based objective function by stochastic gradient descent algorithm. The embedding extraction method relies on maximizing the likelihood of preserving network neighborhood. To achieve this goal, 2^{nd} order biased random walks are used to create network neighborhoods of the nodes.

Given a graph G=(V,E), node2vec algorithm seeks to learn a mapping function $f : u \rightarrow \mathbb{R}^d$ which maximizes the log-likelihood objective function:

$$\max_f \sum_{u \in V} \log Pr(N_s(u)|f(u))$$

where $N_s(u)$ is network neighboorhod of node u. With a conditional independence assumption:

$$\log Pr(N_s(u)|f(u)) = \sum_{n_i \in N_s(u)} \log Pr(f(n_i)|f(u))$$

By use of symmetry in feature space or softmax parametrization:

$$\log Pr(f(n_i)|f(u)) = \frac{exp(f(n_i) \cdot f(u))}{\sum_{v \in V} \exp(f(v) \cdot f(u))}$$

which leads to following objective:

$$\max_{f} \sum_{u \in V} \sum_{n_i \in N_s(u)} \log \frac{exp(f(n_i) \cdot f(u))}{\sum_{v \in V} \exp\left(f(v) \cdot f(u)\right)}$$

This objective function is optimized by SGD by adopting of a negative sampling approach. Since the denominator of the above function is quite expensive to compute, a negative sampling method is used. It is just sampling some of the negative nodes. By this means, only embeddings of a few nodes will be updated at each step while performing the stochastic gradient descent algorithm. Therefore, it will reduce the training time. But at this time, another question arises as to how to determine the network neighborhood $N_s(u)$.

To determine the network neighborhood $N_s(u)$, two traditional strategies, namely breadth-first search (BFS) and depth-first search (DFS), can be considered. BFS is mostly traversing the local neighborhood. In contrast, DFS is an algorithm that explores the network in a global view.

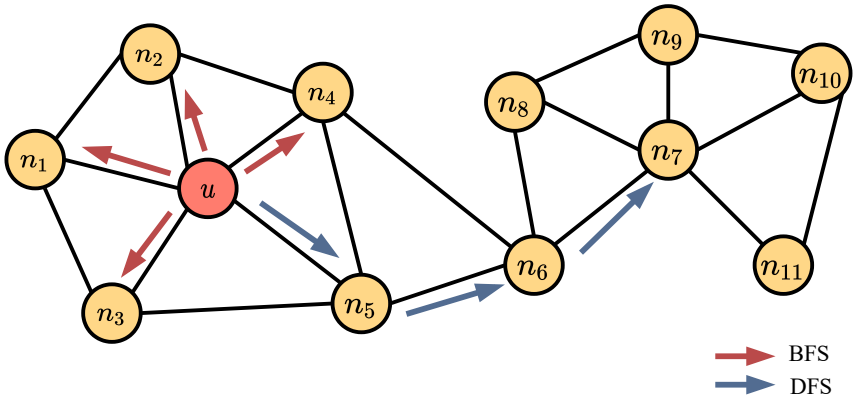

Figure 3.6 BFS and DFS traversing from node u, adopted from [1]

Both traversing strategies has useful aspects for different types of networks to capture neighborhoods. Instead of depending on one traversing strategy, combining both of them by a ratio is the key idea behind node2vec algorithm. By doing so, it takes advantage of both strategies. To this end, in node2vec algorithm there are two parameters that interpolate BFS and DFS: return parameter p (return to the previous node), and in-out parameter q (moving inwards (BFS)-moving outwards(DFS). These parameters are used to perform 2^{nd} order biased random walks around selected node to capture neighborhood.

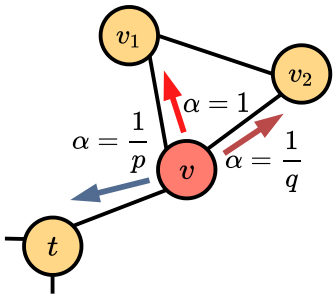

Figure 3.7 Illustration of random walk where walker at node v, adopted from [1]

In Figure 3.7 an illustration of random walk is depicted where a walker at node v, has just came from node t, and the next step is determined by search biases denoted as α.

Formally, a simulation of random walk of fixed length l, a source node u, c_i is the i$^{\text{th}}$ node in the walk, which starts from $c_0 = u$. The following probability distribution is used to generate nodes c_i:

$$P(c_i = x | c_{i-1} = v) = \begin{cases} \frac{\pi_{vx}}{Z}, & \text{if } (v,x) \in E \\ 0, & \text{otherwise} \end{cases}$$

where π_{vx} is the unnormalized transition probability between nodes v and x

To perform a biased random walk, unnormalized transition probabilities are calculated as the multiplication of the static edge weights by a bias factor α. So, when the walker at node v came from node t, the next step at random walk is evaluated based on this probability:

$$\pi_{vx} = \alpha_{pq}(t,x) \cdot w_{vx}$$

where w_{vx} is the static edge weight between node v and node x and α is defined as follows:

$$\alpha_{pq}(t,x) = \begin{cases} \frac{1}{p}, & \text{if } d_{tx} = 0 \\ 1, & \text{if } d_{tx} = 1 \\ \frac{1}{q}, & \text{if } d_{tx} = 2 \end{cases}$$

3.5.2 metapath2vec

metapath2vec algorithm is another powerful network representation learning algorithm to generate node embeddings. It is designed to apply on heterogeneous graphs which contain different type of nodes. To illustrate, for an academic collaboration network, there are two types of nodes which are authors and papers, and the collaboration between them correspond to edges. Since the User-Item Network is also a heterogeneous graph, metapath2vec is worth to be examined for this network.

This method also uses a random walk approach as node2vec and line; in comparison with them, metapath2vec focuses on heterogeneity. The motivation behind this approach is based on the poor evaluation results caused by behaving all the nodes identically, which creates indistinguishable representations for all nodes. It performs meta-path-based random walks to generate latent representations of the networks.

Formally, given a heterogeneous network $G = (V, E, T)$, the mapping functions associated with node v and link e are defined as follows: $\phi(v) : V \to T_V$ and $\phi(e) : E \to T_E$ with $|T_v|$. T_V is the set of node types and T_E is the set of link types, and $|T_V| + |T_E| > 2$. Method learns d-dimensional node embeddings $X \in \mathbb{R}^{|V| \times d}$. Here $\mathbf{X}$ is a matrix and X_v correspond the d-dimensional vector embedding of node v.

metapath2vec algorithm maximizes the network neighborhood probability of corresponding type of nodes that preserves the context information for each node type. Therefore, it optimizes the following objective:

$$arg\max_{\theta} \sum_{v \in V} \sum_{t \in T_V} \sum_{c_t \in N_T(v)} \log p(c_t | v; \theta)$$

where $N_t(v)$ is the network neighborhood of node v with the t^{th} type of nodes, $p(c_t|v;\theta)$ is the conditional probability of context node c_t given node v.
By applying softmax factorization:

$$\log p(c_t | v; \theta) = \frac{exp(X_{c_t} \cdot X_v)}{\sum_{v \in V} \exp(X_u \cdot X_v)}$$

Same as node2vec, negative sampling approach is used for computational efficiency, and this equation gets the following approximation:

$$\log \sigma(X c_t \cdot X_v) + \sum_{m=1}^{M} \mathbb{E}_{u^m \sim P(u)} [\log \sigma(-X_{u^m} \cdot X_v)]$$

where $\sigma(\cdot)$ is sigmoid function, and $P(u)$ is the probability distribution which generates negative samples M times.

In the metapath2vec algorithm, the meta-paths used are used to perform random walks rather than heterogeneous paths, as they tend to highly visible node types that only make up a small portion of nodes.

For a heterogeneous graph $G = (V, E, T)$ and meta path $P : v_1 \xrightarrow{R_1} v_2 \xrightarrow{R_2} \cdots v_t \xrightarrow{R_t} v_{t+1} \cdots \xrightarrow{R_{l-1}} v_l$, where $R = R_1 \circ R_2 \circ \cdots \circ R_l$ defines a composite relation, $\circ$ is composition operator, the transition probability for a random walk at step i is determined as the following:

$$p(v^{i+1}|v_t^i, P)) = \begin{cases} \frac{1}{|N_{t+1}(v_t^i)|} & (v^{i+1}, v_t^i) \in E, \phi(v^{i+1}) = t+1 \\ 0 & (v^{i+1}, v_t^i) \in E, \phi(v^{i+1}, v_t^i) \neq t+1 \\ 0 & (v^{i+1}, v_t^i) \notin E \end{cases}$$

In addition to that, meta-paths are used as symmetrical in the algorithm. Therefore, a recursive approach is adopted to lead the random walk.

$$p(v^{i+1}|v_t^i, P) = 1, \text{ if } t = 1$$

3.6 Link Embeddings

Node embeddings are explained on the above sections in detail but since it is a link prediction task, a "link" formulation is also needed to use in the machine learning framework. An edge/link is the connection between a pair of nodes, but as an input perspective for the prediction task, it is the combination of the embeddings of the nodes that is interacting with each-other by a binary operator.

3.6.1 Binary Operators

A binary operator $\odot$ generates a link representation $g(u, v)$ such that $g : V \times V \rightarrow \mathbb{R}^d$ where d is the vector size based on node embedding vectors $f(u)$ and $f(v)$ of node u and node v. This operator is intended to have a generalization power which also enables to represent non-existent "false" or "negative" links.

In order to preserve vector dimension d in these "link embeddings", three most common binary/element-wise operators are considered in this thesis namely average, hadamard and l2 [1].

Binary Operator	Symbol	Definition
Average	$\boxplus$	$[f(u) \boxplus f(v)]_i = \frac{f_i(u)+f_i(v)}{2}$
Hadamard	$\circ$	$[f(u) \circ f(v)]_i = f_i(u) * f_i(v)$
L2	$\|\cdot\|_2$	$\|f(u) \cdot f(v)\|_{\overline{2}i} = \|f_i(u) - f_i(v)\|^2$

Table 3.1 Binary operators and definitions adopted from [1]

3.7 Data Partition

For binary classification task, three different types of models are considered as classifier. In addition, hyperparameter tuning step is included to select the best model to perform the task. To evaluate the model, data is divided into train, validation, and test sets. Since the task is to predict which item will the user interact in the future time, among the M month period, the first $M-2$ months are used as train, $M-1^{\text{st}}$ month as validation, and the last month in the data set is used as test set to evaluate the models. After the model selection steps, train and validation set are merged to train the best classifier in order to apply on test set and get results because accessing the test set and change the model characteristics according to scores create a bias towards the test set and cause over-fitting which is not the desired behavior. It can cause to get poor results on different test sets.

3.7.1 Train Set

For model evaluation on validation set, positive links in the first M month are used to create a graph (UIN) and train embedding model. For the final evaluation on test data, these sets are merged and used to train the embedding model. After training of embedding model, node embeddings preserving network information are obtained.

Training of the classifier model is achieved by use of both positive and negative links together. Negative links are also important to have a robust classifier that needs to learn a differentiation between existence and non-existence. Positive links are labeled as 1 and negative links are labeled as 0. The process of getting non-existent (negative) links is called negative sampling. Negative sampling is achieved by randomly selecting a pair of nodes that are not linked in the network and generating its link embedding since all node embeddings are present. Applying a binary operator on these vectors gives a non-existing link representation. To have a balance in both classes, number of positive and negative links are selected as equal. Since most real networks are sparse, there are vast numbers of non-existing links. Thus, finding the negative samples is not a complex process.

3.7.2 Validation and Test Set

After embedding and classifier models are trained, validation set and test sets are used for model evaluation. From M month transaction data, validation set and test sets include some users and their $M-2^{\text{nd}}$ and $M-1^{\text{th}}$ month purchase transactions as positive links, respectively. On the other hand, since the task is to discover the newly formed transactions in this month, one should consider all the item combinations for specific user. Simply taking the only positive links of a user for this month is not the right approach because this information is unknown and needs to be discovered.

For a fair evaluation, for one user, all item and this user combinations should be checked where actual transactions are labeled as positive and other combinations as negative links in validation and test sets since they are not included training set. Classifier model should be able to differentiate which links are positive or negative.

3.8 Classifier Model

In order to perform binary classification of true links and false links, a classifier model is needed. For this reason, a couple of machine learning models are considered in this thesis to compare their performances because various machine learning models can give different predictive performances according to applied data sets. Due to high predictive powers of gradient boosting algorithms and artificial neural networks, these models are considered for the binary classification task in this thesis.

3.8.1 Boosting Algorithms

Boosting is introduced by Kearns and Valiant, which relies on the concept of an ensemble of weak learners creating a strong learner model. By mean weak learner, which is the model slightly better than random predictions. Boosting algorithms train a single weak classifier and then train another weak classifier which only focuses on data parts where the first weak classifier got poor results. Again, another weak classifier comes into the picture, which focuses on the parts the second weak learners got wrong. This process is going on iteratively. In the end, a whole strong learner may be produced from multiple weak learners. Adaboost [50] is the first robust algorithm based on boosting approach, which will be examined in this section.

Boosting algorithms have some similarities with random forest algorithms, but the core procedure is different. Random forest algorithms use independent fully grown tree models with low bias and high variance as learners; in the end, it combines the predictions according to averaging and majority voting principles for different parts of data to reduce variance. In boosting, this combining stage is done in a sequential way in contrast to random forest. Contribution weights of the weak learners depend on the prediction performance of each model. In this method, wrong predictions have increased weights in order to be picked up with a high probability in the next round. Therefore, there is an iterative process that does not emerge a clear independence.

3.8.1.1 Gradient Boosting Algorithms

Gradient boosting [51] is a generalized version of boosting algorithms that uses gradient descent for optimization. It also has characteristics of boosting algorithms, such as combining weak learners to form a strong one. In comparison with Adaboost explained in the previous section, gradient boosting algorithms use a residual concept in their methodology. After each round, residuals are used as input to the next weak learner. In the end, the residuals are combined with a loss function to get an overall loss. The objective is to obtain a model which minimizes this overall loss.

Formally, given a training data $\{x_i, y_i\}_{i=1}^{N}$ where $x_i = (x_{1i}, x_{2i}, \cdots, x_{di})$, y_i is the label for i^{th} data point, N is the total number of data points. The main aim is to find a optimized classification function $F(x)$ which minimizes the loss function $L(y_i, F(x_i))$ as follows:

$$F^*(x) = arg\min_{F} L(y_i, F(x_i))$$

Since this equation has an infinite number of parameters in function space, this is approximated by the following solution, which is designed as additive form:

$$F^*(x) = \sum_{m=1}^{M} f_m(x)$$

where $f_0(x)$ is initialized value and $\{f_m(x)\}_1^M$ are incremental formed functions defined as "boosts".

The model is initialized with a constant value γ as follows:

$$F_0(x) = arg\min_{\gamma} \sum_{i=1}^{N} L(y_i, \gamma)$$

where $L(y_i, \gamma)$ is loss function
Gradient function is defined as:

$$\hat{r_i} = -\left[\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)}\right]_{F(x)=F_{m-1}(x)}$$

where m corresponds to the number of iterations and $m = 1, 2, \cdots, M$
For steepest descent $f_m(x)$ is defined as follows:

$$f_m(x) = \gamma_m r_m(x)$$

$T(x_i; \alpha_n)$ is the base learner with parameters α_n and β_n that characterizes this function [51, 72];

$$F(x; \{\beta_m, \alpha_m\}_1^M) = \sum_{m=1}^{M} \beta_m T(x; \alpha_m)$$

After fitting of data, to approximate $F^*(x)$, by use of approximator $F_{m-1}(x)$, the function $\beta_m T(x; \alpha_m)$ can be seen as the best greedy step which is highly correlated with negative gradient direction (residuals $\hat{r_i}$) and can be obtained by optimization:

$$\alpha_m = arg\min_{\alpha,\beta} \sum_{i=1}^{M} [\hat{r_i} - \beta T(x_i; \alpha)]^2$$

This $T(x_i; \alpha)$ function is used in place of negative gradient in steepest descent method [51]. By performing a line search [73] the following approximation is obtained for the model weights γ:

$$\gamma_m = arg\min_{\gamma} \sum_{i=1}^{M} L(y_i, F_{m-1}(x) + \gamma T(x_i; \alpha_m))$$

Model update is achieved by the following:

$$F_m(x) = F_{m-1}(x) + \gamma_m T(x_i; \alpha_m)$$

This iterative process goes on until convergence or iteration times are reached.

3.8.1.2 XGBoost

XGBoost [12] (eXtreme Gradient Boosting) is a framework based on gradient boosting algorithm. It introduces regularization that reduces overfitting. Due to parallelized and distributed structure of XGBoost, it also reduces training time.

By adding a regularization term the objective function used in XGBoost is as follows:

$$O = \sum_{i=1}^{N} L(y_i, F(x_i)) + \sum_{m=1}^{M} \Omega(f_m) + C$$

where the first term corresponds to training loss measuring of how well model fit on training data and $\Omega(f_k)$ is regularization or penalty term at m time iteration measuring complexity of trees, C is a constant and M is the number of trees. By optimizing training loss model, model gets more predictive nature which is close to training data. In addition to that, optimizing regularization gives simpler models which make more stable predictions with reduced variances. The objective is to balance these terms and obtain a model both predictive and simple [74].

Regularization term $\Omega(f_k)$ is defined as:

$$\Omega(f_k) = \alpha H + \frac{1}{2}\lambda \sum_{j=1}^{H} w_j^2$$

where α is complexity parameter of leaves which can be considered as a measure of how much split occurs in tree, H is the number of leaf nodes, λ is regularization or penalty parameter and w_j is the output of each leaf node in tree. So, the complexity of tree can be expressed in terms of the number of leaf nodes and prediction scores of each leaf node.

Second-order Taylor series expansion of objective function is used with a mean square error loss function. Hence, the objective function becomes the following [74]:

$$O = \sum_{i=1}^{n} [g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i)] + \Omega(f_t)$$

g_i and h_i are first and the second derivatives of the loss function. The tree is redefined by the scores of leaf nodes. Therefore, a function $q(x_i)$ is defined, which maps each data points to appropriate leaves, and the following mapping is used:

$$f_t(x) = w_{q(x)}$$

where w is the output scores of leaf nodes and t is boosting round. After this redefinition, the objective function becomes the following:

$$O = \sum_{i=1}^{n} [p_i w_{q(x_i)} + \frac{1}{2}(q_i w^2_{q(x_i)})] + \alpha H + \frac{1}{2}\lambda \sum_{j=1}^{H} w_j^2$$

Leaves are representing the data samples, and the loss values at these nodes can be summed up and used as the final loss value. After this process objective function can also be written in the following form:

$$O = \sum_{j=1}^{T} [P_j w_j + \frac{1}{2}(Q_j + \lambda) w_j^2] + \alpha H$$

where $P_j = \sum_{i \in I} p_i$, $Qj = \sum_{i \in I} q_i$, and I_j indicates all samples in leaf node j as I_j. Hence, the problem becomes simply as minimization of a quadratic function. Moreover, by adding regularization, overfitting is reduced by this model.

3.8.1.3 LightGBM

Although gradient boosting algorithms have very satisfying performances in machine learning tasks, because of increase in data sizes in recent years, traditional gradient boosting decision tree (GBDT) algorithms have some challenges about efficiency while working on big data. In order to overcome these challenges, feature selection methods are used by calculating the information gain in the decision tree splits to examine feature importance. Another approach is decreasing data size and features.

LightGBM [13] brings a new idea into gradient boosting approach as Gradient-based One-Side Sampling (GOSS). In this method, the data points which cause larger gradient changes are kept more while data sampling in order to have more information gain.

Exclusive Feature Bundling (EFB) is another technique that is newly introduced by LightGBM. They regroup the mutually exclusive features into bundles and take them as a single feature which reduces the number of features.

3.8.2 Artificial Neural Networks

As a classifier model, Artificial Neural Networks (ANN) are also considered for the prediction task. In this section, very brief background information about ANN will be given.

A neural network is an information processing system that is modeled based on biological neural networks. An artificial neuron is the smallest unit of cell that carries information in these systems. Actually, this information can be thought of as holding a number. These neurons are fired up (being activated) when the given threshold values are reached, which has some similarities between real neurons, which have action potentials that start their activation according to electrical voltages.

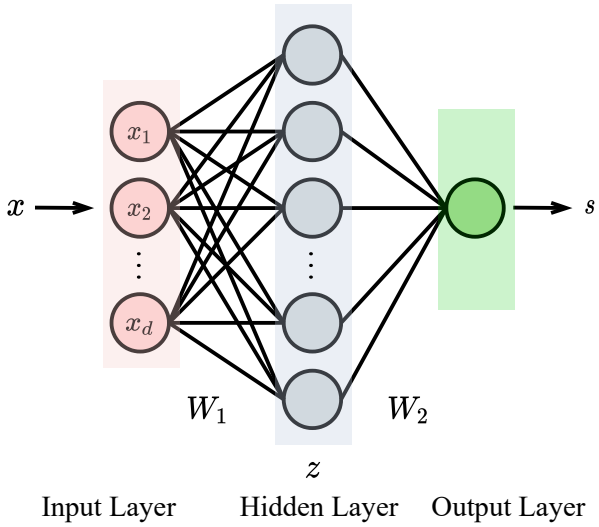

Figure 3.8 Feed Forward Neural Network.

An artificial neural network tries to learn a function approximation that maps input values to output values. This function approximation is achieved by combining neuron-containing layers. There are weights that are multiplied with input data and convey information to other layers. After weight multiplication, a non-linear function such as sigmoid, reLu, tanh is applied to incoming values. A feed-forward neural network (only considered neural network architecture in this thesis) performs one pass after these steps and calculates a loss based on a specified loss function since the labels are known in train data. After passing these steps, gradient descent algorithm is applied on loss function for minimization.

Given a training data $\{x_i, y_i\}_{i=1}^{N}$ where $x_i = (x_{1i}, x_{2i}, \cdots, x_{di})$, y_i is the label for i^{th} data point, N is the total number of data points. Non-linarities σ_1 and σ_2 are used as ReLu and sigmoid function where $\sigma_1(x) = \max(0, x)$ and $\sigma_2(x) = \frac{1}{1+e^{-x}}$.

$$z = \sigma_1(W_1 x)$$

$$s = \sigma_2(W_2 z)$$

where W_1 is layer 1 weights and W_2 is layer 2 weights. Loss function L is chosen as binary cross-entropy for binary classification:

$$L = -\frac{1}{N}\sum_{i=1}^{N} y_i \log(s_i) + (1-y_i)\log(1-s_i)$$

Gradient descent algorithm is used based on the following update rules for optimization.

$$W_2^{old} = W_2^{new} - \eta \frac{\partial L}{\partial W_2}$$
$$W_1^{old} = W_1^{new} - \eta \frac{\partial L}{\partial W_1}$$

To obtain these partial derivatives, chain rule is used. For the layer 2 weights:

$$\frac{\partial L}{\partial W_2} = \frac{\partial L}{\partial s}\frac{\partial s}{\partial W_2}$$

$$\frac{\partial L}{\partial s^{(i)}} = \frac{1}{N}\cdot\frac{-y_i}{s^{(i)}} + \frac{(1-y_i)}{(1-s_i)}$$

$$\frac{\partial s^{(i)}}{\partial W_2^{(i)}} = z\cdot\sigma_2'(W_2 z) = z\cdot\sigma_2(W_2 z)\cdot(1-\sigma_2(W_2 z)) = \frac{z^{(i)}e^{-W_2^{(i)}z^{(i)}}}{(1+e^{-W_2^{(i)}z^{(i)}})^2}$$

$$\frac{\partial L}{\partial W_2^{(i)}} = \left(\frac{1}{N}\cdot\frac{-y_i}{s^{(i)}} + \frac{(1-y_i)}{(1-s_i)}\right)\cdot\left(\frac{-z^{(i)}e^{-W_2^{(i)}z^{(i)}}}{(1+e^{-W_2^{(i)}z^{(i)}})^2}\right)$$

For the layer 1 weights:

$$\frac{\partial L}{\partial W_1} = \frac{\partial L}{\partial s}\frac{\partial s}{\partial z}\frac{\partial z}{\partial W_1}$$

$$\frac{\partial s^{(i)}}{\partial z^{(i)}} = W_2 \cdot \sigma_2'(W_2 z) = W_2 \cdot \sigma_2(W_2 z) \cdot (1 - \sigma_2(W_2 z)) = \frac{W_2^{(i)} e^{-W_2^{(i)} z^{(i)}}}{(1 + e^{-W_2^{(i)} z^{(i)}})^2}$$

$$\frac{\partial z^{(i)}}{\partial W_1^{(i)}} = x \cdot \sigma_1'(W_1 x) = \begin{cases} x, & \text{if } W_1 x > 0 \\ 0, & \text{otherwise} \end{cases}$$

Then;

$$\frac{\partial L}{\partial W_1} = \begin{cases} \left(\frac{1}{N} \cdot \frac{-y_i}{s^{(i)}} + \frac{(1 - y_i)}{(1 - s_i)} \right) \cdot \frac{W_2^{(i)} e^{-W_2^{(i)} z^{(i)}}}{(1 + e^{-W_2^{(i)} z^{(i)}})^2} \cdot x, & \text{if } W_1 x > 0 \\ 0, & \text{otherwise} \end{cases}$$

3.9 Alternative Method

In order to perform a comparative analysis, a method is considered that is used as a common solution to the research task. This problem is addressed with a recommendation system. This is a common task in e-commerce, movie recommendations and much more.

3.9.1 Collaborative Filtering

Collaborative filtering is a common data mining method that is used in recommendation systems. For a recommendation system, collaborative filtering technique generates a matrix where each row represents a user and each column is assigned to an item. The value in their intersection is usually corresponds to rating value which the user has given for this item. This matrix representation can be adapted to various concepts such as user-movie recommendation, user-product recommendation, or user-song recommendation systems.

Implicit and Explicit Data sets

The data which is gathered from the users can be classified as implicit or explicit. In explicit data sets, there is rating information which may be a rating of a movie in a range of 1-5 given by a user. On the other hand, in explicit data sets, there is not a clear rating information. Instead, there may be some user actions that can be used to recommend items to users. These actions can be number of times a song is played, an item purchase, how many times a customer visits a merchant

to make a purchase. Therefore, one can say that if a person plays a song so many times than the other songs, this user likes this song. These implicit signals are used in many recommendation systems. Most transactional data sets (such as purchase data) do not contain rating information; therefore, they are implicit data sets.

3.9.2 Alternating Least Squares

In this thesis, a latent factor model type of collaborative filtering is used as an alternating method. The number of times a user interacts with an item is used as the implicit information. Alternating Least Squares (ALS) method is a case of matrix factorization technique.

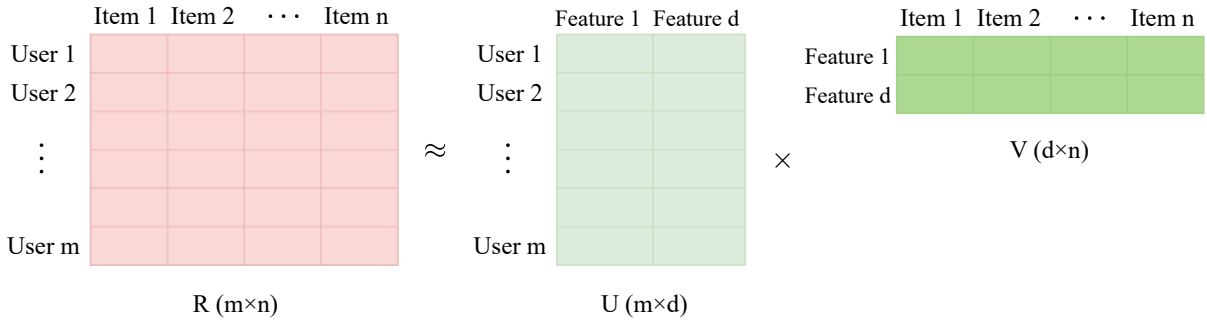

Figure 3.9 Matrix Factorization of interaction matrix R into U as user features and V item features matrix.

For implicit data sets, an interaction matrix R is defined which represents user and item interactions. The main idea behind ALS method is to factorize this interaction matrix R into smaller matrices U as user features, and V item features. r_{ui} which is the interaction value of user u with item i, which can be number of times a song played or number of times an item is added to the cart etc. Preference p_{ui} of user u on item i is calculated based on the interaction value r_{ui} as follows:

$$p_{ui} = \begin{cases} 1 & r_{ui} > 0 \\ 0 & r_{ui} = 0 \end{cases}$$

According to this equation, if a user u has interacted with item i, it is believed that this user has liked this item. Otherwise, it is considered as not having an interest. However, these interactions may be misleading. For example, a user may be unaware of the existence of an item that may be interested. Similarly, an item may have been purchased as a present for a friend of the user. Therefore, a confidence value c_{ui} is introduced, which can be defined as how much value a user u gives for that item i.

$$c_{ui} = 1 + \alpha r_{ui}$$

By this equation, larger confidence is obtained on user-item pairs having higher interaction values r_{ui}. Also, this confidence value increases linearly by a factor of α.

A user-factors vector $x_u \in \mathbb{R}^f$ for user u, item-factors vector $y_i \in \mathbb{R}^f$ for item i. Dot product is used to create a prediction for missing data as follows:

$$\hat{r}_{ui} = x_u^T y_i$$

To find the optimal user interaction and item interaction matrices, the following cost function is used:

$$\min_{y_*, y_*} \sum_{u,i} c_{ui} \left(p_{ui} - x_u^T y_i \right)^2 + \lambda \left(\sum_u \|x_u\|^2 + \sum_i \|y_i\|^2 \right)$$

where λ is regularization factor determined by cross validation. Since this cost function has $m \cdot n$ terms, which reach billions, stochastic gradient descent algorithm cannot be applied. Therefore, an alternating approach is adopted for optimization. By fixing user and item interaction matrices and taking derivatives alternatively, cost function becomes quadratic, which can be optimized by alternating-least-squares method. This process continues iteratively, and at each step cost function is minimized. Taking derivatives of cost function gives the following equations:

$$x_u = \left(Y^T C^u Y + \lambda I \right)^{-1} Y^T C^u p(u)$$

$$y_i = \left(X^T C^i X + \lambda I \right)^{-1} X^T C^i p(i)$$

Then, these equations can also be broken into the followings that reduce computational complexity:

$$x_u = \left(Y^T Y + Y^T \left(C^u - I \right) Y + \lambda I \right)^{-1} Y^T C^u p(u)$$
$$y_i = \left(X^T X + X^T \left(C^i - I \right) X + \lambda I \right)^{-1} X^T C^i p(i)$$

By using them, randomly initialized X and Y are alternatively updated and reduce cost function.

Chapter 4

Implementation

In this section, proposed method implementation details will be given. First, the whole architecture is written in Python [75] programming language due to its high flexibility. Google Colab [1] which is a cloud-based Jupiter notebook service provided by Google is used as the coding platform. This platform also provides GPU support that accelerates training process.

4.1 Embedding Extraction

4.1.1 node2vec Implementation

Stellargraph framework is providing a node2vec implementation that is also based on Gensim, which is a Python package for natural language processing. Gensim package has a Word2vec [44] implementation. Since node2vec is an adaptation of Word2vec in a network perfective, all parameters in Word2vec are also valid for node2vec. Here is the list of parameters related to node2vec, their descriptions, and their value ranges.

Parameter	Value Range	Description
p	$[0, 2]$	how probable returns to source node
q	$[0, 2]$	how much it explore the network
dimensions	$[32, 100]$	embedding dimension
num_walks	$[1, 100]$	number of random walks done for each root node
walk_length	$[50, 200]$	length of a random walk from a source root
num_iter	$[1, 10]$	number of SGD epochs

Table 4.1 node2vec paramater list

[1] https://colab.research.google.com/

Algorithm 1: node2vec

Function `LearnFeatures(`*Graph* $G(E,V,W)$*, Dimensions* d*, Number of walks* r*, Walk length* l*, Context size* k*, Return* p*, In-out* q`):`

```
π = PreprocessWeights(G, p, q)
G' = (V, E, π)
walks ← empty
for i ← 0 to r do
    foreach u ∈ V do
        walk = node2vecWalk(G', u, l)
        Add walk to walks
    end
end
f = StochasticGradientDescentOptimization(k, d, walks)
return f
```

Function `node2vecWalk(`*Graph* $G(E,V,W)$*, Starting node* u*, Walk length* l`):`

```
walk ← [u]
for i ← 0 to l do
    curr = walk[−1]
    Vcurr = GetNeighbors(curr, G')
    s = AliasSample(V_curr, π)
    Add s to walk
end
return walk
```

4.1.2 metapath2vec

Stellargraph [2] framework also provides a metapath2vec implementation which is similar to node2vec but different in meta-path concepts. Since node2vec and metapath2vec are both based on Word2vec, all parameters in Word2vec are also valid for metapath2vec. The list of parameters related to metapath2vec, their descriptions, and their value ranges are given below.

Parameter	Value Range	Description
dimensions	$[1,10]$	embedding dimension
metapath	–	meta path scheme
num_walks	$[1,100]$	number of random walks done for each root node
context_window_size	$[1,10]$	context window size of Word2Vec
walk_length	$[50,200]$	length of a random walk from a source root
num_iter	$[1,10]$	number of SGD epochs

Table 4.2 metapath2vec paramater list

[2] http://stellargraph.readthedocs.io

Algorithm 2: metapath2vec

Input : Heteregenous Graph $G(E,V,T)$, Meta-Path P, number of walks per node w, walk length l, Dimension d, Neighborhood size k

Output: Node embedding matrix $\mathbf{X} \in \mathbb{R}^{|V| \times d}$

for $i \leftarrow 1$ **to** w **do**
  foreach $v \in V$ **do**
    $MP = MetaPathRandomWalk(G, P, v, l)$
    $X = HeteregenousSkipGram(X, k, MP)$
  end
end
return X

Function `MetaPathRandomWalk(`*Heteregenous Graph $G(E,V,T)$, Meta-path P, Node v, Walk length l*`):`
  $MP[1] = v$ **for** $i \leftarrow 1$ **to** $l-1$ **do**
    $u = NegativeSampling()$
    $MP[1] = u$
  end
  return MP

Function `HeteregenousSkipGram(`*Node Embedding Matrix X, Neighborhood size k, Meta-path P*`):`
  $walk \leftarrow [u]$
  for $i \leftarrow 1$ **to** l **do**
    $v = MP[i]$
    for $j = \max(0, i-k)$ **to** $\min(i+k, l)$ ***and*** $j \neq i$ **do**
      $c_t = MP[j]$
      $X = StochasticGradientDescent(X, v, c_t)$
    end
  end
  return $walk$

4.2 Classifier Model

4.2.1 XGBoost

XGBoost [12] provides implementation in various environments such as Python, R, and JAVA. Python version is used in this thesis. Some important parameters of XGB package are listed as follows:

Parameter	Value Range	Description
eta	$[0,1]$	Learning rate
max_depth	$[0,20]$	Maximum depth of the decision tree
max_bin	$[16,256]$	Maximum # discrete bins to collect continuous features
num_rounds	$[0,1000]$	Number of boosting rounds

Table 4.3 XGBoost paramater list

4.2.2 LightGBM

LightGBM, a gradient boosting framework developed by Microsoft, is designed to be distributed and efficient with lower training times and memory usage, higher efficiency. It also provides GPU support that can work with large-scale data.

Parameter	Value Range	Description
eta	$[0,1]$	Learning rate
max_depth	$[0,20]$	Maximum depth of the decision tree
is_unbalance	{true,false}	Checks the data set unbalanced or not
feature_fraction	$[0,1]$	Random selection data without resampling
begging_fraction	$[0,1]$	Random selection a subset of features at each round
num_rounds	$[0,1000]$	Number of boosting rounds

Table 4.4 LightGBM paramater list

4.2.3 Artificial Neural Networks

Keras [3] provides an Tensorflow [76] based easy-to-use, scalable neural network framework in Python environment. The parameters which can be used in Keras framework are listed below.

Parameter	Value Range	Description
hidden_units	$[0,256]$	Number of neurons at a layer
input_dimension	$[0,20]$	Size of the input
optimizer	{SGD,RMSprop,Adam}	Optimizing method at backpropagation
loss	$[0,1000]$	Loss function
lr	$(0,0.1]$	Learning rate
batch_size	$[16,128]$	# samples that are used at each update
activations	{relu,sigmoid,softmax,tanh}	Activation function

Table 4.5 ANN paramater list

Repository

All codes that are used to implement proposed system are freely available at a GitHub repository. [4]

[3] https://keras.io/

[4] https://github.com/alaattinyilmaz/link-prediction-recommender

Chapter 5

Use Case: Merchant Prediction

In this chapter, a use case scenario of proposed link prediction model will be presented with a real-life transaction data set. Throughout this chapter, problem formulation of merchant prediction task, the motivation behind using this scenario with proposed model will also be examined.

5.1 Problem Formulation

Merchant prediction refers to discovering the potential merchants where a customer can visit and make a purchase. Transaction data has these customer-merchant pair information which can be used to predict future purchases. In this context, link prediction based methodology developed in this thesis is used to discover merchants where a customer can make a purchase but not visited before by this customer. These predictions can be used as recommendations.

5.2 Motivation

Due to advancements in information technologies, the availability of customer transaction data has raised [77]. Hence, this gives an opportunity to have more insights into consumer purchase behavior, which plays a huge role in determining marketing strategies. Moreover, it is also useful for detecting the target market, which has many advantages on increasing sales.

Detecting the audience can help merchants to create personalized recommendations, advertising campaigns, promotions, and allocate resources efficiently [78, 79]. In classical methods, audience demographics extracted from census data, psychographics and customer surveys are used for segmentation of customers to target

audience [80]. In addition to these statistics, product categories or market-basket analysis methods are also used to understand customer purchase preferences. However, their ability of customer purchase prediction is limited [81].

To overcome their limitations, probability modeling approaches [77] that capture latent purchase behavior characteristics from transaction data are adopted. Some methods combine probability distributions (e.g., Poisson, binomial, exponential) to characterize user behaviors [77]. In addition to these approaches, machine learning methods such as gradient boosting tree, random forest algorithms combined with customer features are giving convincing performances [79]. Moreover, collaborative filtering based methods such as Alternating Least Squares (ALS) for implicit feedback data [14] which is mainly used on item recommendation, can also be used for merchant prediction task.

In this context, the motivation behind predicting the next month's merchants is to detect the audience for personalized advertising and allocate resources efficiently.

5.3 Data set

A transactional credit card data was provided by a private bank from an OECD country and used throughout the research. In order to preserve privacy, data set is anonymized, which does not allow one to identify the users and merchants. Although the data is anonymized, the assigned anonymous identification numbers for customers and merchants are used as the same values for the entire data set to ensure consistency.

MUSTERI_ID_MASK	ISLEM_TARIHI	ISLEM_SAAT	ISLEM_TUTARI	UYEISYERI_ID_MASK	X	Y	mcc_cat
11664741	2015-01-22	12:24:27	309.35	69846423	41.03265	28.99068	Doğrudan Pazarlama- Sigorta Hizmetleri
2949796	2015-01-03	11:06:54	274.99	69846423	41.03265	28.99068	Doğrudan Pazarlama- Sigorta Hizmetleri
17360000	2015-01-03	14:30:24	873.1	69846423	41.03265	28.99068	Doğrudan Pazarlama- Sigorta Hizmetleri
11871471	2015-01-03	15:37:48	411.26	69846423	41.03265	28.99068	Doğrudan Pazarlama- Sigorta Hizmetleri
11595923	2015-01-03	12:42:32	468.12	69846423	41.03265	28.99068	Doğrudan Pazarlama- Sigorta Hizmetleri

Table 5.1 Snapshot from the transaction data set

Each row corresponds to a user purchase transaction, and the columns represent customer id, operation time, transaction amount, merchant id, latitude and longitude coordinates of merchant location and merchant category, respectively.

In this credit card transaction data, there are 627184 transactions made by 51451 customers from 482 different merchants with a time span of July 2014 to June 2015. The vast majority of transactions are made in Istanbul, a major city in Turkey.

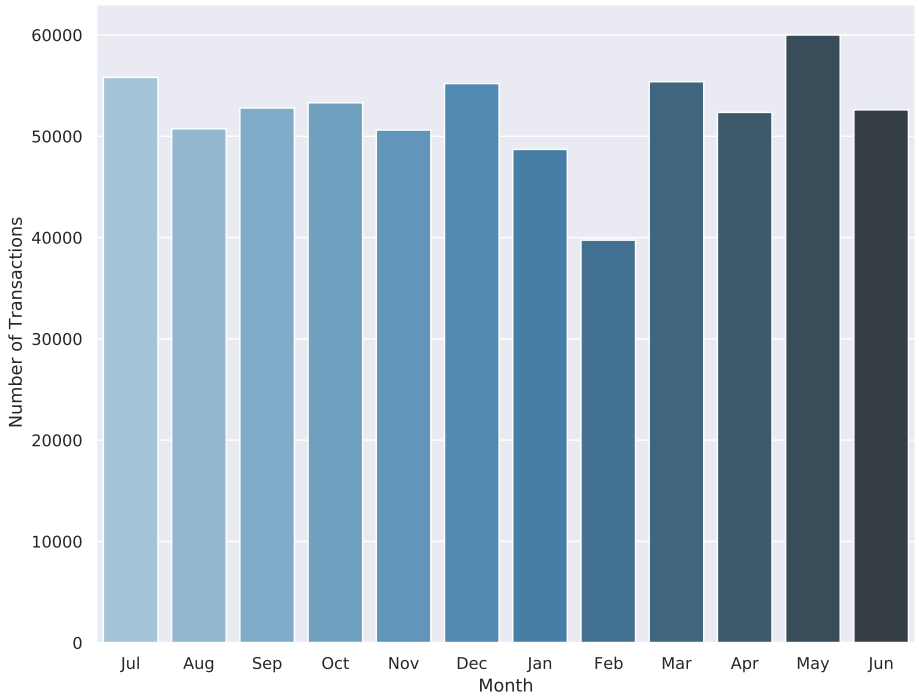

Figure 5.1 Number of transactions according to months

Each merchant has a merchant category and these are as follows: Service Stations, Houseware Shops, Grocery Stores and Supermarkets, Direct Marketing and Insurance Services, Man and Woman Clothing Shops, Electronics Shops, Other Commercial Activities, Public Services, Telecommunication Services, Sports Shops, Insurance Sale, Car Parks and Garages, Food Places and Restaurants, Airlines, Travel Agents and Tour Operators, Hospitals, Cosmetics Shops, Bus Routes.

5.4 Merchant-Customer Network

Data Cleaning and Prepocessing

The transaction data set is preprocessed to create Merchant-Customer Network where nodes are customers and users, and links represent the purchase occurrence between them.

At preprocessing step, some merchant categories are discarded, which may create noise while training the models. After dropping these categories, only merchants that are under the following categories are taken into consideration throughout the thesis: Grocery Stores and Supermarkets, Electronics Shops, Man and Woman Clothing Shops, Houseware Shops, Service Stations, Sports Shops, Food Places, and Restaurants.

Network Visualization

The sampled visualization of MCN can be seen below, where red nodes represent customers, and yellow nodes correspond to merchants. As the whole network is really huge, sampled version is used for better visualization.

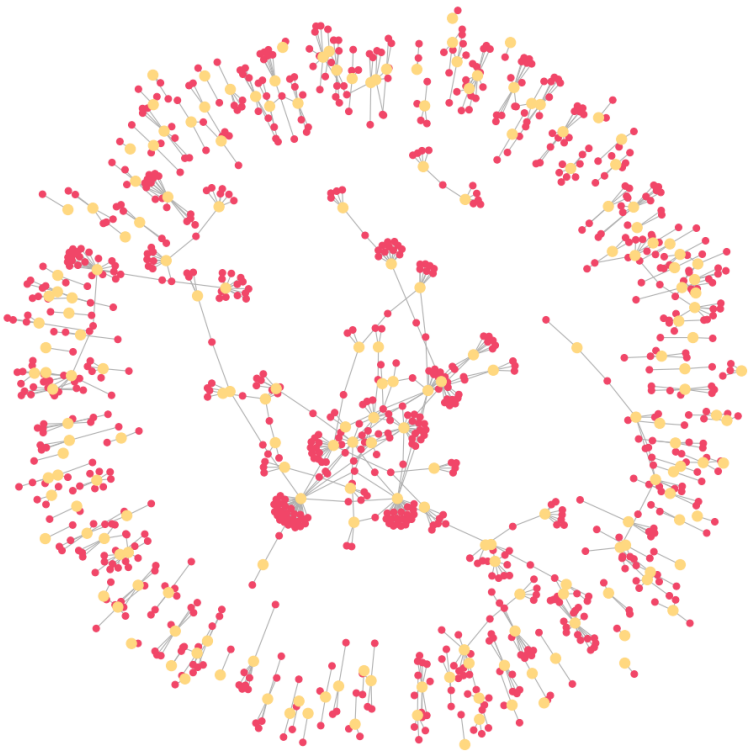

Figure 5.2 Sampled visualization of Merchant-Customer Network.

In Figure 5.3 unique-merchant customer pair distribution can be seen. According to that bar graph, Service Stations, and Grocery Stores, and Supermarkets categories are dominant at transaction numbers. In Figure 5.4 the number of merchants are represented according to their merchant categories. If both figures 5.3 and 5.4 are considered together, it can be interpreted that Houseware Shops category has dense connections since it has a relatively lower number of merchants but a higher number of transactions. Furthermore, Figure 5.5 shows the degree distribution of the network in log scale gives information about the probability of a node chosen randomly has a degree of k.

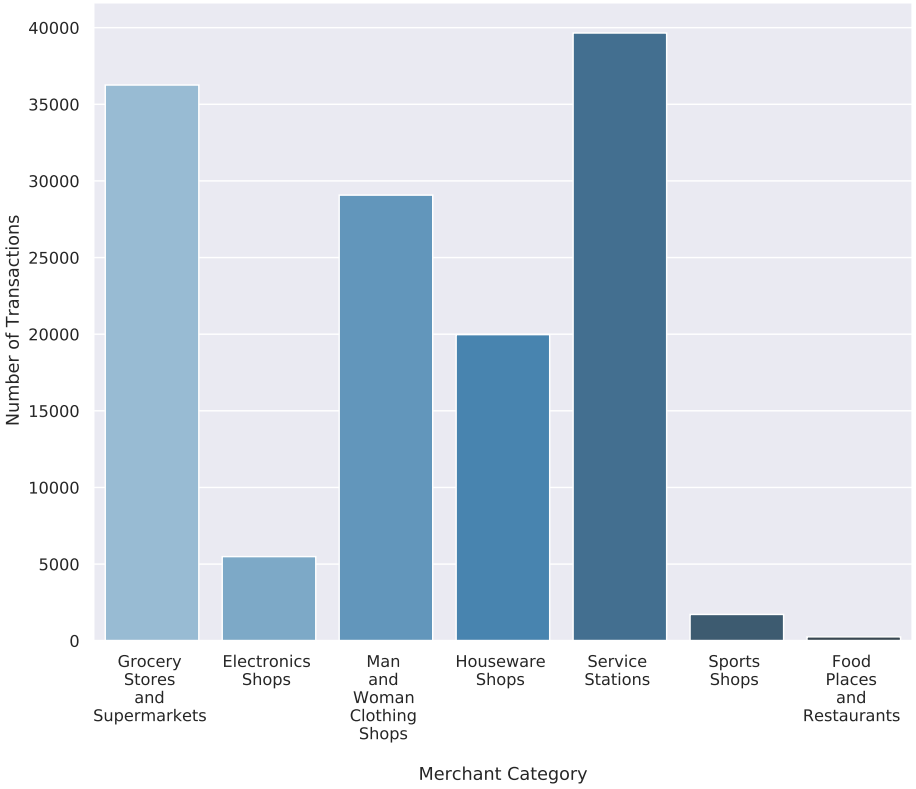

Figure 5.3 Unique merchant-customer pair distribution according to merchant category

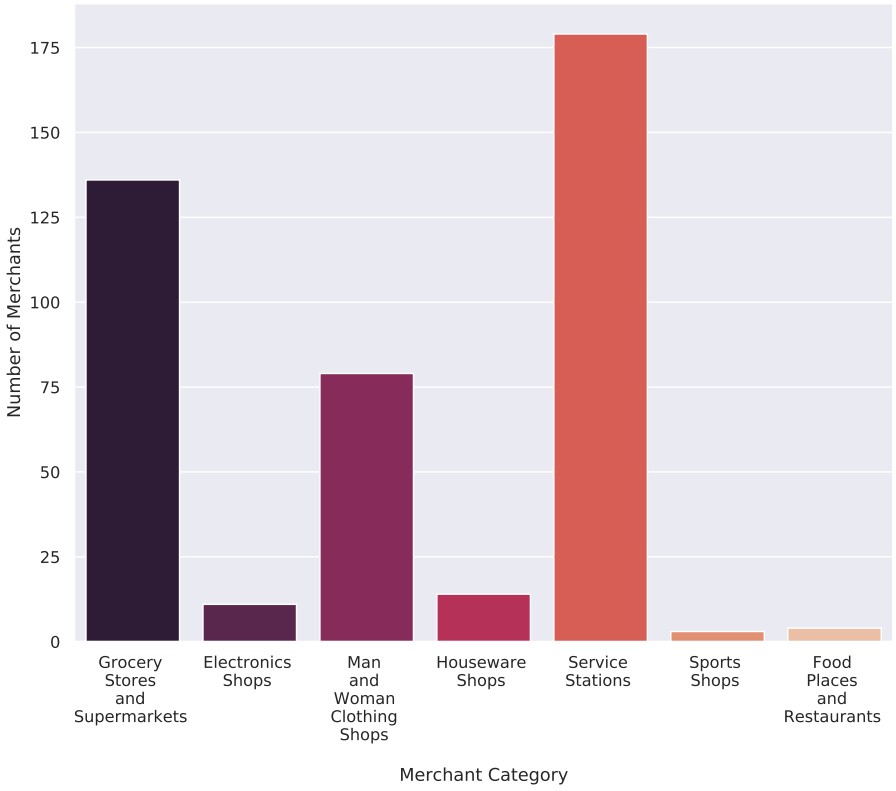

Figure 5.4 Number of merchant distribution according to merchant category

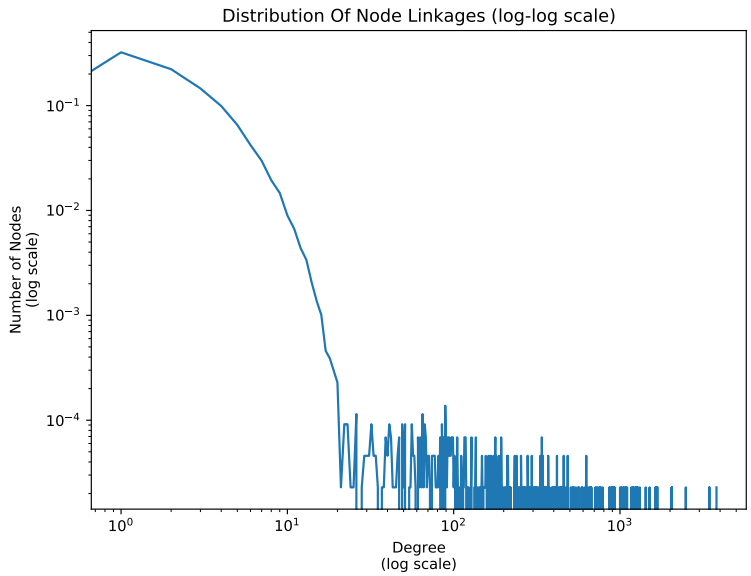

Figure 5.5 Degree distribution of MCN in log scale

5.5 Data Partition

As defined in proposed model architecture, data is splited as train, validation and test sets. Train set contains transactions from the first 10 months, the validation set has the 11^{st} month, and test set has the 12^{nd} month as positive links. In addition to them, an equal number of negative transactions are randomly generated and added to train set.

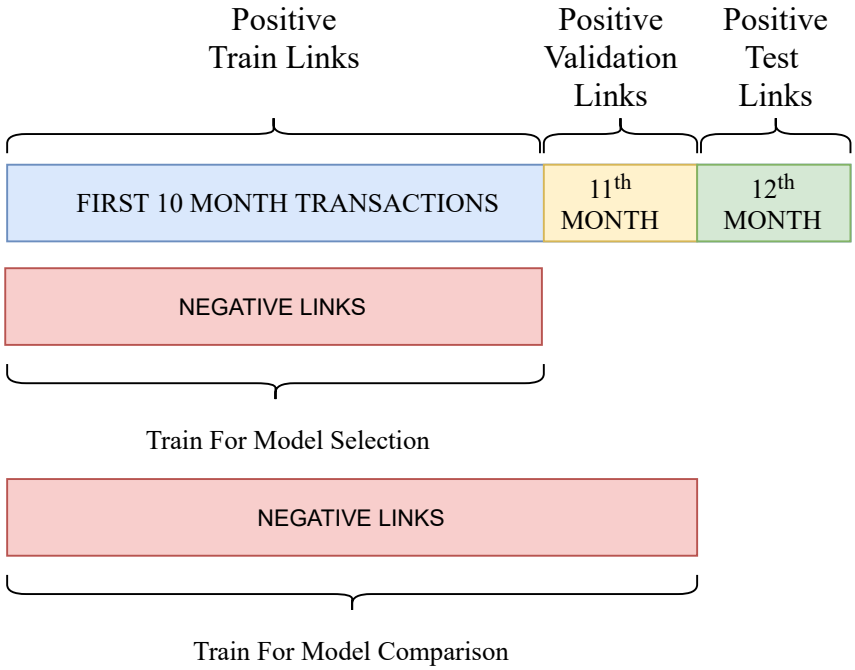

Figure 5.6 Data Partition Visualization

On the other hand, for test and validation sets, since the task is to discover the newly formed transactions on these months, in addition to positive links, all merchant-customer combinations are considered for the specific customer as negative links. Simply taking the only positive links of a customer would not be an accurate approach, because this information is not known and has to be detected by proposed model.

Data	**# Pos. Links**	**# Neg. Links**	**# Customers**	**# Merchants**
Train	119931	119931	41431	397
Validation	8410	5777942	14723	397
Train+Validation	132410	132410	43245	426
Test	7165	5743713	13640	426

Table 5.2 Data partition details

5.6 Embedding Visualization

Node embeddings extracted by node2vec and metapath2vec are visualized by projecting into 2-dimensional space by t-SNE [43] dimensionality reduction method. By doing so, it can give a better understanding of underlying characteristics that forms clusters in the embedding space.

Link embeddings are generated by applying binary operator on node embeddings of pairs of nodes. These embeddings can also represent negative links which is the link that does not exist between two nodes. Similarly, node2vec and metapath2vec based link embeddings according to average binary operators are generated and visualized. Due to the hugeness of data, PCA [41] dimensionality reduction technique is used instead of t-SNE in link embedding visualization.

In Figure 5.7 and 5.8 metapath2vec and node2vec node embeddings are visualized. Moreover, due to maintain flow of the thesis, town labeled node embedding and link embeddings visualizations are presented in Appendix A.

Node Embeddings

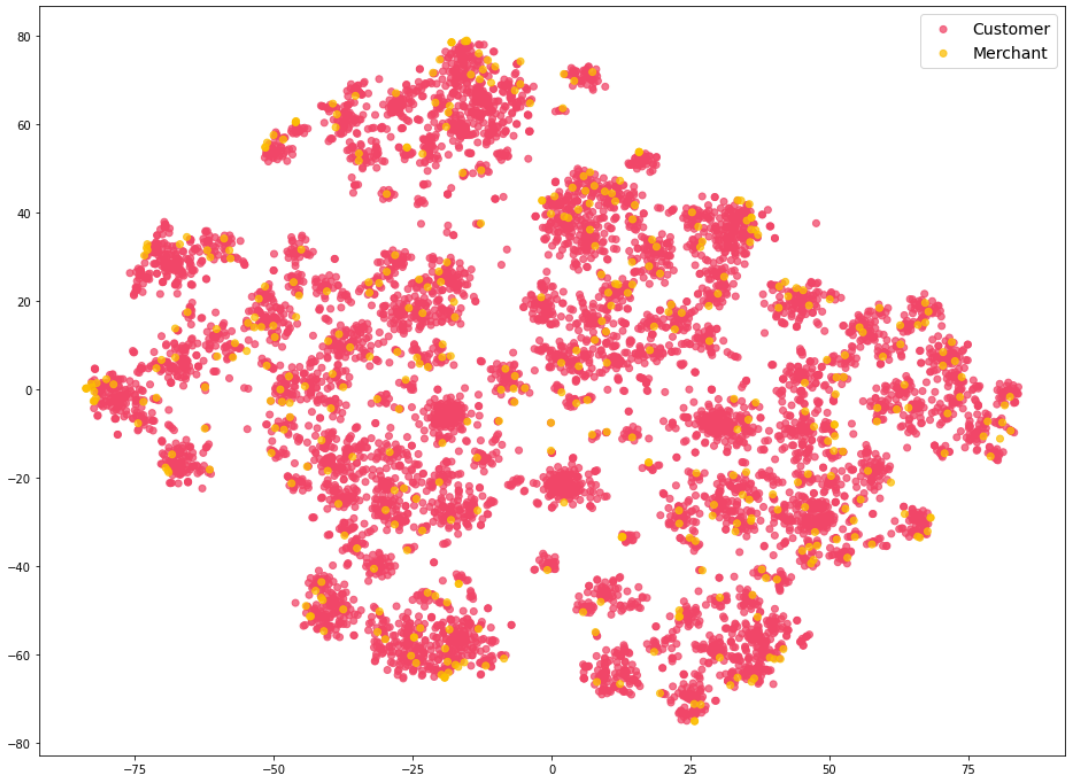

Figure 5.7 Metapath2vec Sampled Node Embedding Visualization

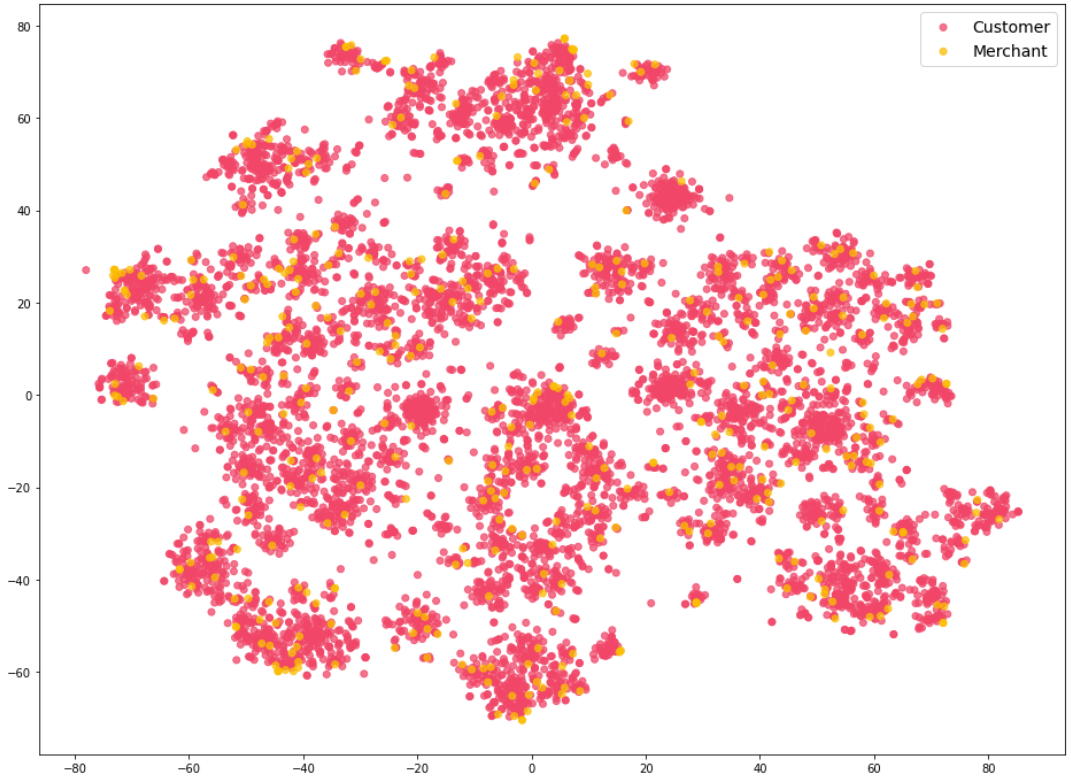

Figure 5.8 Node2vec Sampled Node Embedding Visualization

Chapter 6

Evaluation

In order to decide which methods mentioned in architecture design are better, an evaluation process is performed. This process consists of embedding model, classifier model and binary operator selection stages. In addition, hyper-parameter tuning is conducted for each model.

6.1 Evaluation Metrics

Deciding on a "goodness" of a recommendation provided by a recommendation system is based on some evaluation metrics. To evaluate and compare performances of models, receiver operating characteristic (ROC) curve, mean average precision at K (MAP@K) and area under curve (AUC) metrics are used in this thesis.

6.1.1 Receiver Operating Characteristic

For classification problems, Receiver Operating Characteristic curve is widely used. It plots the true positive rate against the false positive rate for different threshold values. A classifier is considered good when its ROC curve lies up the y-axis and along the x-axis. However, poor classifier ROC curve lies on x=y line at the coordinate system. Area Under Curve (AUC) is, as its name implies, the area under the ROC curve and a numerical value between 0 and 1.

6.1.2 Mean Average Precision at K

Mean Average Precision at K (MAP@K) metric treats the recommendation system as a ranking task since recommendation systems offer a ranked list of recommendations based on their recommendation score. This score can be similarity, probability, or any other measurements.

MAP@K is a common metric that is widely used in information retrieval and recommendation systems. It aims to evaluate how relevant a recommendation is.

$$AP@K = \frac{\sum_{k=1}^{K} P(k) \times rel(k)}{R}$$

where AP@K is average precision at K, R is the number of relevant documents, $rel(k)$ is an indicator function equals to 1 if the item at rank k is relevant, 0 otherwise.

$$MAP@K = \frac{\sum_{q=1}^{Q} AP@K_q}{Q}$$

where Q is the number of queries, AP@K$_q$ is average precision at K value for this specific query q.

In merchant prediction context, Q is the number of customers being considered at validation or test set, R is the number of new purchases that need to be discovered for a customer. AP@K value is calculated for each user and taking the average of these values gives MAP@K value.

6.1.3 Mann-Whitney U-Test

Mann-Whitney U-Test [82] is a non-parametric test that examines the null hypothesis as probability of X is greater than Y and probability of Y is greater than X are equal where X and Y are random samples selected from two populations.

Formally, Let $X_1, \cdots, X_n$ be an independent and identically distributed (i.i.d.) sample from X, and $Y_1, \cdots, Y_m$ an i.i.d. sample from Y, X and Y are statistically independent of each other. Then, U-Test statistics are defined as follows:

$$U = \sum_{i=1}^{n} \sum_{j=1}^{m} S(X_i, Y_j)$$

$$S(X,Y) = \begin{cases} 1, & \text{if } Y < X \\ \frac{1}{2}, & \text{if } Y = X, \\ 0, & \text{if } Y > X. \end{cases}$$

For link prediction evaluation of merchant prediction task, prediction scores are divided as positive and negative link groups, and samples taken from these groups will be used in the U-Test. This test will examine whether positive link scores are significantly higher than negative link scores or not.

6.2 Experiments

To evaluate the effectiveness of proposed model, many number of experiments are conducted. In the first experiments, the training set (the first 10 months of transactions) and validation set (11^{st} month transactions) are used to decide which embedding model, binary operator and combination is the best. There are a total of 72 different combinations in the experiments. After the decision, the best combination is used to compare with alternative method which is the Alternating Least Squares (ALS) on test data. Training data used in comparison consists of merging of train and validation set.

Embedding Models	**Classifer Models**	**Binary Operators**
node2vec	XGBoost	Average
metapath2vec	LightGBM	Hadamard
	ANN	L2

Table 6.1 Summary of models and operators to be selected

6.2.1 Embedding Model Hyperparameter Tuning

Number of walks and walk length are the predominant hyper-parameters that affect embedding model performances. Therefore, the effects of different values of these hyperparameters are examined. For other configurations, meta-path used in metapath2vec model is customer → merchant → customer and embedding dimension is 100. There are 8 different embedding models present in the experiments.

Model Name	**Embedding Type**	**Number of walks**	**Walk length**
N1	node2vec	1	100
N2	node2vec	1	200
N3	node2vec	10	100
N4	node2vec	10	200
MP1	metapath2vec	1	100
MP2	metapath2vec	1	200
MP3	metapath2vec	10	100
MP4	metapath2vec	10	200

Table 6.2 Embedding Model hyperparameters

6.2.2 Classifier Model Configurations

In order to decide classifier model, aforementioned classifier models with default parameter values are considered in the selection process.

Model Name	Parameters
LGBM	objective: binary, is_unbalance: true, feature_fraction: 0.5, bagging_fraction: 0.5, bagging_freq: 20, num_boost_round=500
XGB	max_depth: 11, eta:0.3, objective: binary logistic, max_bin:16, num_rounds: 500
ANN	hidden_units: 32, input_dimension:100, optimizer: adam, loss: binary cross entropy, activations: reLu, sigmoid

Table 6.3 Classifier Model parameter configurations

6.3 Results

6.3.1 Validation Results

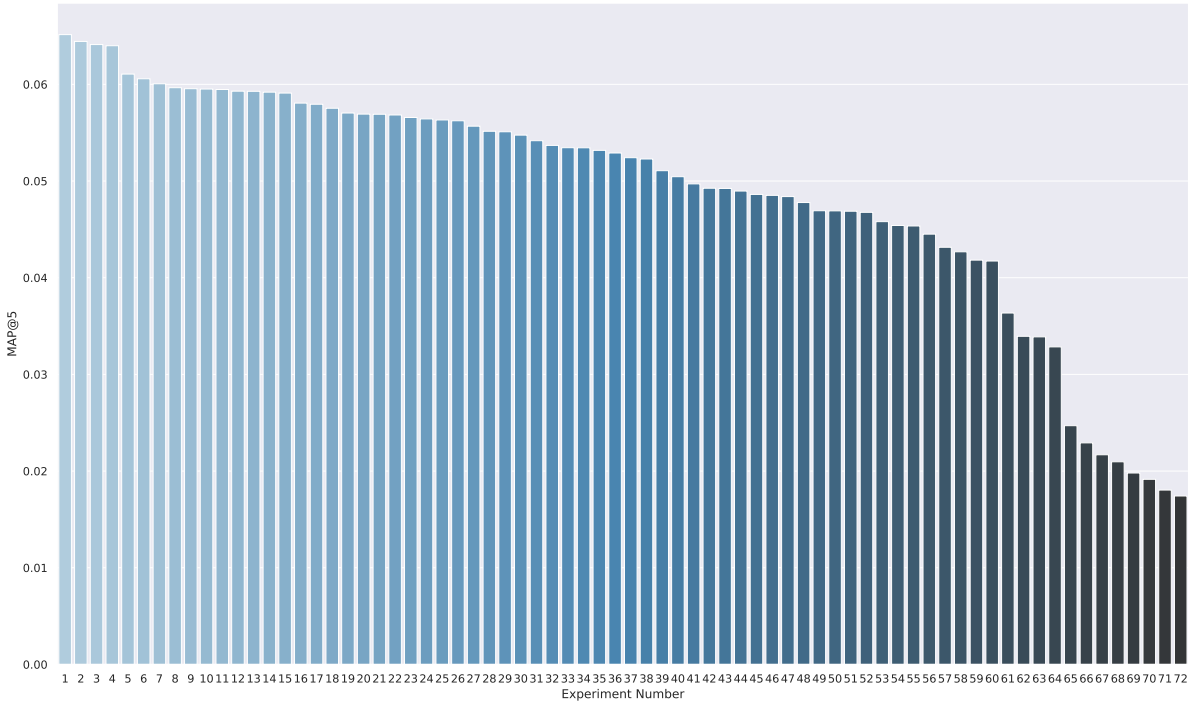

Figure 6.1 MAP@5 Scores according to experiment numbers (see Table 6.4 and 6.5)

Classifier Model

Table 6.4 and 6.5 show the experiment results ordered by MAP@5 values on validation set. Among 72 experiments, the best MAP@5 score is 0.065144 with LGBM + metapath2vec + average operator. LightGBM classifier model is dominant at the first 10 ordered experiments. XGBoost classifier is the second which is shown until 16^{th} experiment. ANN comes into picture after 18^{th} experiment that shows the worst performance among classifiers.

According to results, it can be interpreted that LGBM creates more complex trees that give better results in comparison with XGBoost. Also, ANN prediction scores are close to each other in general, which means they are mostly accumulated near 1 or 0, which does not create a powerful, distinguishing feature for a ranking task.

Rank	Classifier	Embedding Model	Operator	AUC	MAP@5
1	LGBM	MP2	avg	0.842366	0.065144
2	LGBM	MP1	avg	0.867941	0.064438
3	LGBM	N1	avg	0.86557	0.064115
4	LGBM	N2	avg	0.83629	0.063995
5	LGBM	N1	hadamard	0.860167	0.061063
6	XGB	N1	avg	0.85034	0.060578
7	LGBM	MP2	hadamard	0.85298	0.060059
8	XGB	MP2	avg	0.833313	0.059653
9	LGBM	MP1	hadamard	0.859542	0.05955
10	XGB	MP1	hadamard	0.845396	0.059514
11	XGB	MP1	avg	0.851308	0.059469
12	XGB	N1	hadamard	0.844432	0.059295
13	XGB	MP1	l2	0.843222	0.059279
14	XGB	N1	l2	0.839112	0.059192
15	XGB	N2	avg	0.828527	0.059097
16	XGB	N2	hadamard	0.825707	0.05806
17	ANN	MP3	l2	0.858522	0.057942
18	XGB	MP2	hadamard	0.836183	0.057528
19	ANN	N3	l2	0.859616	0.057033
20	LGBM	N2	hadamard	0.844917	0.056921
21	ANN	MP1	hadamard	0.848236	0.056906
22	ANN	N1	hadamard	0.849119	0.056831
23	ANN	N1	avg	0.873278	0.056572
24	XGB	MP2	l2	0.852815	0.056424
25	XGB	N2	l2	0.846407	0.056317
26	LGBM	N1	l2	0.856985	0.056233
27	XGB	MP3	l2	0.844462	0.055673
28	ANN	MP4	l2	0.855041	0.055147
29	XGB	N3	l2	0.849336	0.055095
30	ANN	N4	l2	0.850823	0.054748
31	LGBM	N2	l2	0.862117	0.054171
32	ANN	MP2	avg	0.864014	0.053685
33	LGBM	MP2	l2	0.864245	0.053445
34	LGBM	MP1	l2	0.857637	0.053436
35	XGB	MP4	l2	0.847255	0.053172
36	ANN	MP2	l2	0.857006	0.052898

Table 6.4 Validation experiment results part 1

Embedding Model

For embedding model selection, models with metapath2vec embeddings show better performance in the first 2 experiments. Although models with node2vec embeddings have convincing results in the next experiments, due to heterogeneous nature of the network built in this thesis, it is not surprising that models with metapath2vec embeddings results are better. Also, walk length on heterogeneous

Rank	Classifier	Embedding Model	Operator	AUC	MAP@5
37	ANN	MP1	avg	0.869216	0.052412
38	XGB	N4	l2	0.840477	0.052278
39	ANN	MP2	hadamard	0.83607	0.051069
40	ANN	N2	l2	0.851234	0.050448
41	ANN	N2	avg	0.861046	0.049704
42	ANN	N2	hadamard	0.825901	0.049255
43	ANN	MP1	l2	0.842776	0.049232
44	ANN	N1	l2	0.841221	0.048963
45	LGBM	MP4	avg	0.851825	0.048614
46	LGBM	N3	avg	0.839608	0.048518
47	LGBM	MP3	avg	0.839772	0.048391
48	ANN	N4	avg	0.859196	0.047775
49	ANN	MP3	avg	0.854452	0.046927
50	LGBM	N4	avg	0.847614	0.046913
51	ANN	MP4	avg	0.855556	0.046877
52	ANN	N3	avg	0.855993	0.046762
53	XGB	MP3	avg	0.831444	0.045798
54	XGB	MP4	avg	0.845583	0.045403
55	XGB	N3	hadamard	0.833318	0.04535
56	XGB	N3	avg	0.83273	0.044507
57	XGB	MP3	hadamard	0.830757	0.043149
58	XGB	MP4	hadamard	0.835488	0.042679
59	XGB	N4	hadamard	0.833846	0.041819
60	XGB	N4	avg	0.843175	0.041724
61	ANN	MP3	hadamard	0.831502	0.036346
62	ANN	N3	hadamard	0.832816	0.033939
63	ANN	MP4	hadamard	0.830758	0.03388
64	ANN	N4	hadamard	0.830609	0.032843
65	LGBM	N3	hadamard	0.841425	0.024693
66	LGBM	MP3	hadamard	0.830858	0.022915
67	LGBM	MP3	l2	0.828677	0.021685
68	LGBM	MP4	hadamard	0.823009	0.020958
69	LGBM	N4	hadamard	0.828977	0.019798
70	LGBM	N3	l2	0.832715	0.019147
71	LGBM	N4	l2	0.817684	0.018035
72	LGBM	MP4	l2	0.821527	0.017422

Table 6.5 Validation experiment results part 2

networks seems to be an important hyper-parameter by examining orderings of MP1 and MP2. In most experiments, MP2 has a superior performance (has higher orders) than MP1 while fixing other features. This situation is opposite in node2vec embeddings (for homogeneous networks); while comparing N1 and N2 embeddings, N1 has superior performance than N2. Furthermore, increasing number of walks can cause overfitting and make the performance worse based on these results.

Binary Operator

Among the binary operators, average operator is the best which dominates the first 4 experiments with the highest MAP@5 values. Hadamard operator is the second best operator after average based on the presence in the first ordered experiments. L2 operator has the worst performance which is dominating the last 3 experiments. Since the node embeddings are representing coordinate values in the latent space, it is a reasonable approach to take element-wise averages in a geometrical perspective that preserving information of both nodes to form a link.

There are only small changes on AUC scores throughout the experiments. Moreover, there is an unbalance at test set between positive and negative links where positive links are only the small portion of entire test set. Because of this, model can predict the negative values more easily that increase AUC score. Therefore, it can be interpreted that AUC scores are not so significant measure to differentiate the validation models. However, this variation is high at MAP@5 values. Based on these observations, experiments are ordered based on their MAP@5 values.

6.3.2 Test and Comparison Results

Proposed and alternative models are compared according to MAP@5, AUC scores. Also, U-Test statistics are calculated in order to examine the significance of the results.

Model	AUC	MAP@5	Mean U-Test Statistics
Proposed	0.8557	0.064718	331.421
Alternative	0.6954	0.047719	278.145

Table 6.6 Comparisons of proposed and alternative methods

Receiver Operating Characteristic

According to AUC scores and ROC curves, proposed model has shown a superior performance than the alternative model. This can also be interpreted from the shapes of the ROC curves. ROC Curve of proposed model is above than the alternative method, and more lied on y-axis and x-axis which shows a closer shape to perfect classifier than alternative method.

Since the prediction scores are in alternative method in cosine similarity form which is between -1 and 1, these scores are scaled between 1 and 0 by use of min-max normalization method to plot ROC curve.

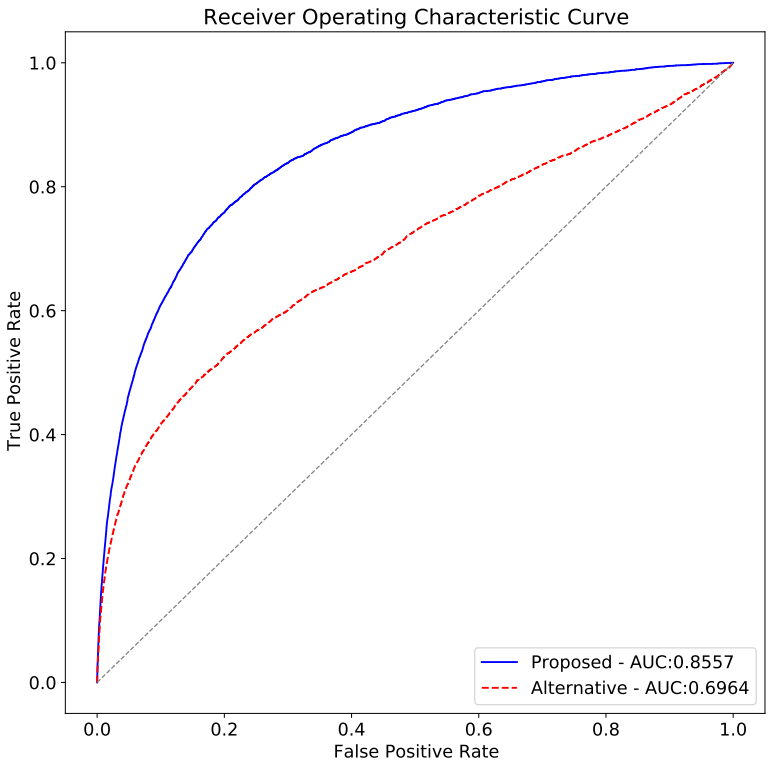

Figure 6.2 ROC Curve and AUC score of test set applied on proposed and alternative models

MAP@K

Higher MAP@K results show that the recommendation system can make more relevant recommendations. Figure 6.3 shows the MAP@K score for different K values from 1 to 5. Naturally, increase of K recommendation will also increase MAP@K result. For K is 5, MAP@K values are 0.064718 and 0.047719 for proposed and alternative methods, respectively. According to these values, proposed model has shown 35% superior performance than alternative method. Even both models recommends the same merchants in their top-5 recommendations, proposed method recommends the merchants in the higher ranks compared to alternative method which is measured by MAP@K.

U-Test Statistics

Mann-Whitney U-Test is applied in order to evaluate proposed and alternative method performances. It is tried to show that prediction scores for positive links are significantly higher than negative link predictions. Since every combination of merchants and customers is considered in the test set, there are a huge number of negative links. To perform a reliable significance test, sampling is applied. The test procedure is as follows: among the test customers, 10 people are selected randomly, then positive and negative links of these people taken. Positive links are constitutes

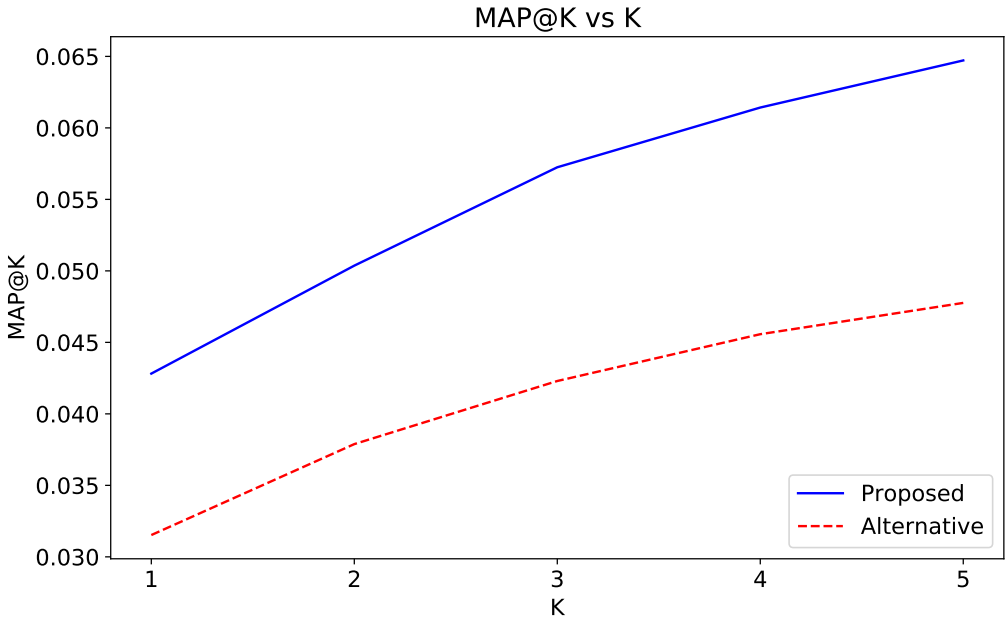

Figure 6.3 MAP@K Comparison of proposed and alternative models

the positive class directly and 50 samples are randomly selected from negative links of these people and named as negative class. U-Test is applied between positive class and sampled negative class of these 10 people. This whole process (including taking 10 people randomly) is repeated 1000 many times.

Model	**Mean U-Test Stats.**	**Std U-Test Stats**
Proposed	331.421	6.9475
Alternative	278.145	12.6452

Table 6.7 Significance comparisons of proposed and alternative methods

According to results presented at 6.5, p-values of both methods are distributed in smaller ranges than 0.05, which means they have significant results. Since p-value distribution of proposed method is in smaller ranges, and its U-Test statistics are higher than alternative method, proposed method has more significant results. This means that positive link prediction scores are significantly higher than negative prediction scores, and obtaining these predictions by chance has a small probability. Also, this probability is lower at proposed method than alternative.

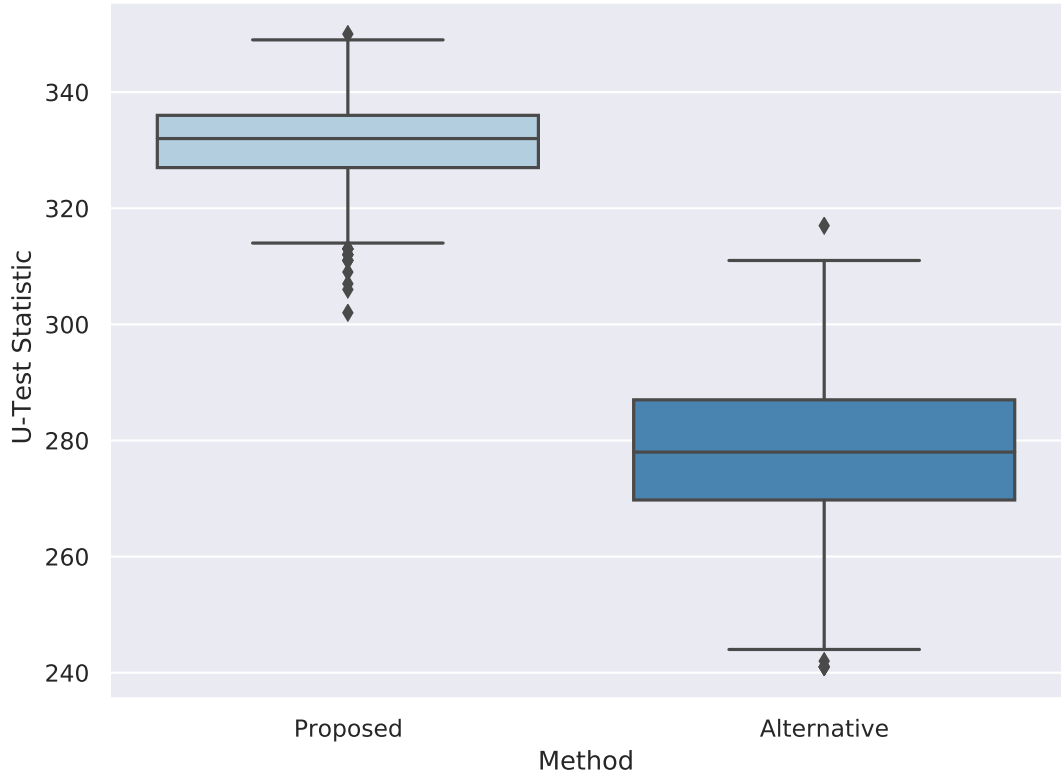

Figure 6.4 U-Test statistic distribution of proposed and alternative method

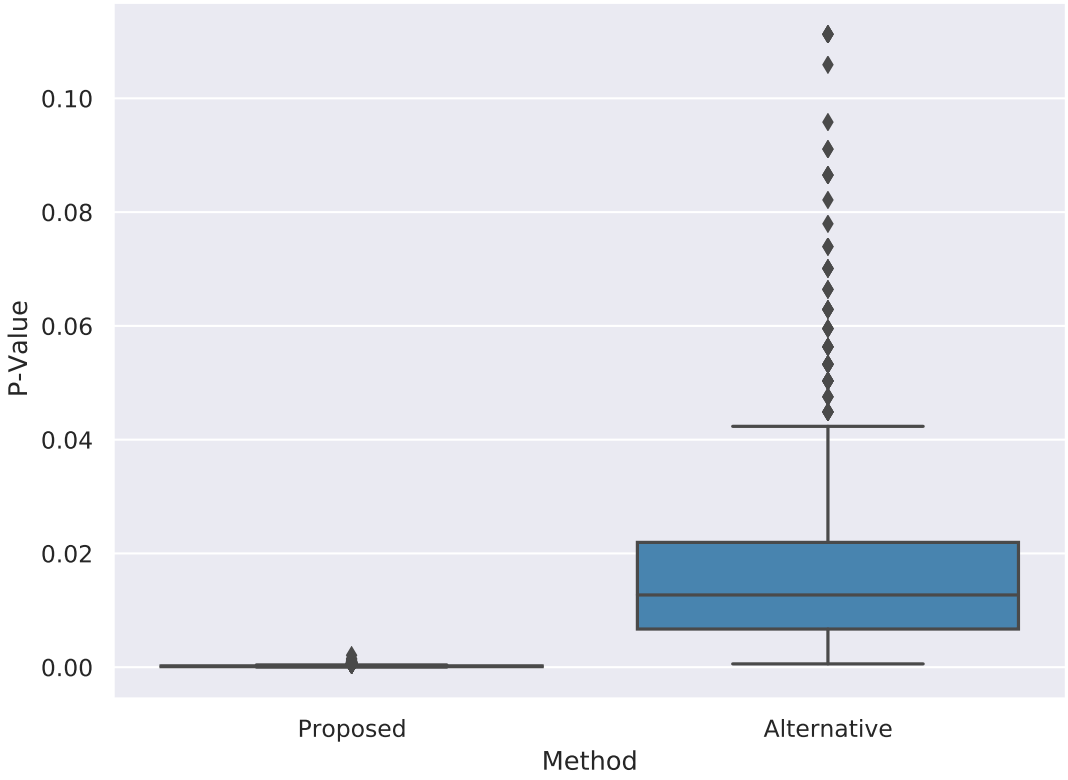

Figure 6.5 P-Value distribution of proposed and alternative method

Detection Rates

Models are also compared based on their detection rates for 5 recommendations. It is the ratio of detected merchant purchases/visits in the top-5 recommendations to total merchants purchase/visits of two models.

		Proposed Model		
		Catch	Miss	Total
Alternative Model	Catch	0.1495	0.0708	0.2203
	Miss	0.1546	0.6251	0.7797
	Total	0.3041	0.6960	

Table 6.8 Catch and Miss rates of top-5 recommendations of proposed and alternative methods

Proposed method has caught 30.41% of the merchant purchases for this month compared to alternative method as 22.03% in their top-5 recommendations.

Map Comparisons

Recommendations of both models for some sample users are visualized on the map. These visualizations include top-5 recommendations of both models and the ground truth merchant visits of a sample customer. Ground truth merchants are shown as markers and 5 recommendations denoted as R#rank (e.g. R1 is the first recommendations, R2 is the second) are shown as circles (blue for proposed method, red for alternative method).

Catch-Miss

In Figure 6.6, proposed model has caught 3 of 3 actual visits (named as GT: ground truth) in the top-5 recommendation merchants as $R2$, $R3$ and $R5$. 2 other merchant recommendations are also close to ground truths geographically. However, in Figure 6.7 alternative method could not catch these merchants and make other recommendations. R2 can be considered as close to a ground truth merchant but R1, R4 and R5 are far away from the actual visit places.

Catch-Catch

In Figure 6.8, proposed model has caught 2 of 3 ground truths (GT) in the top-5 recommendation merchants as $R1$ and $R3$. Again, 3 other merchant recommendations are also close to GTs geographically. Alternative method has caught 1 GT as the 4^{th} recommendation, which can be seen in Figure 6.9. R1 can be considered as close to a ground truth merchant, but R3 is far away from the actual visit places.

Miss-Miss

There are also some instances where both models fail to catch ground truths. In Figure 6.10, proposed model missed the GTs, but its recommendations R1, R2, R4, and R5 are close to GTs. Only R3 is far away from the GTs. Similarly, alternative method has also missed the merchants visits which can be seen in Figure 6.11, its recommendations are far away except R2 in comparison with proposed model.

Miss-Catch

In this case, alternative method has caught 2 of 3 ground truths as R1 and R2 which can be seen in Figure 6.13. However, proposed model could not catch any GTs for this case (see Figure 6.12). Although it fails to detect the GTs, its recommendations R2,R3 and R5 are so close to 2 GTs. Also, R4 and R1 recommendations of proposed model are closer to another GT, which alternative method could not catch, and its recommendations are not far away from that GT in comparison with R3, R4, R5 recommendations of alternative method.

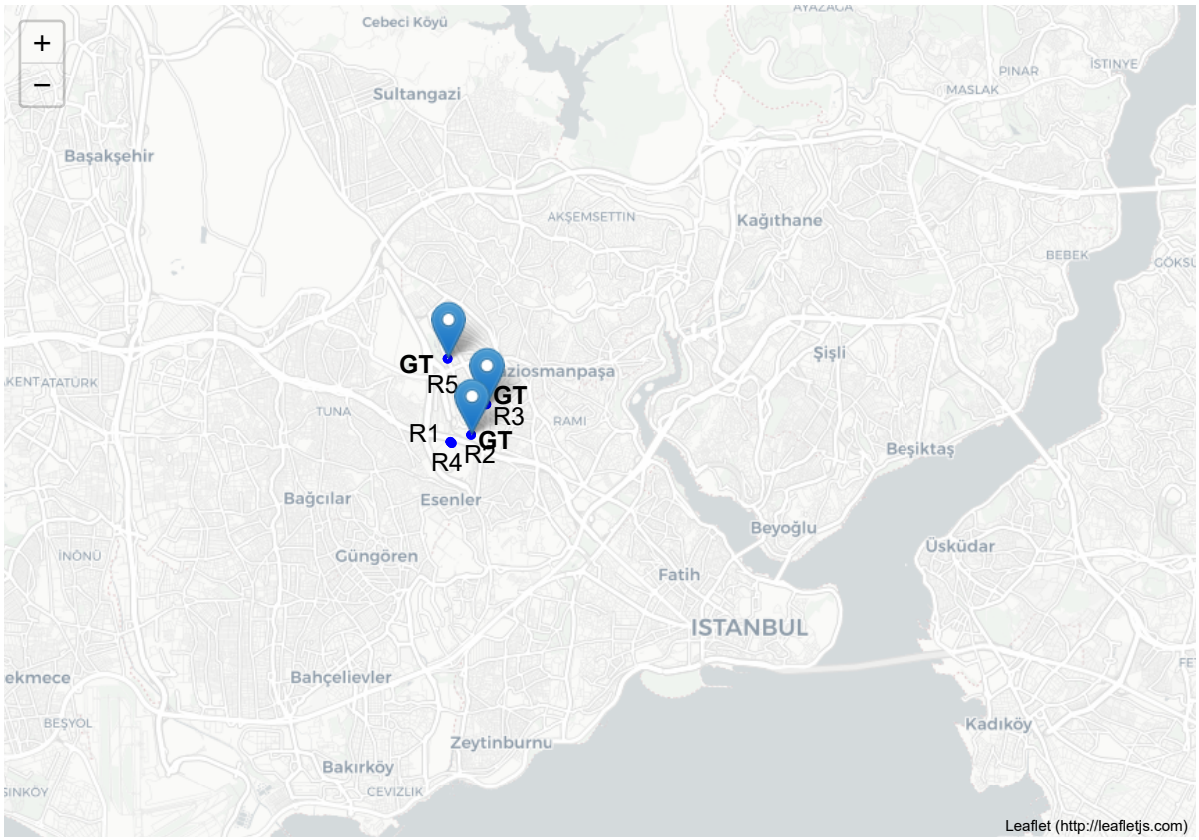

Figure 6.6 Map Visualization of proposed method catching recommendations for a sample user

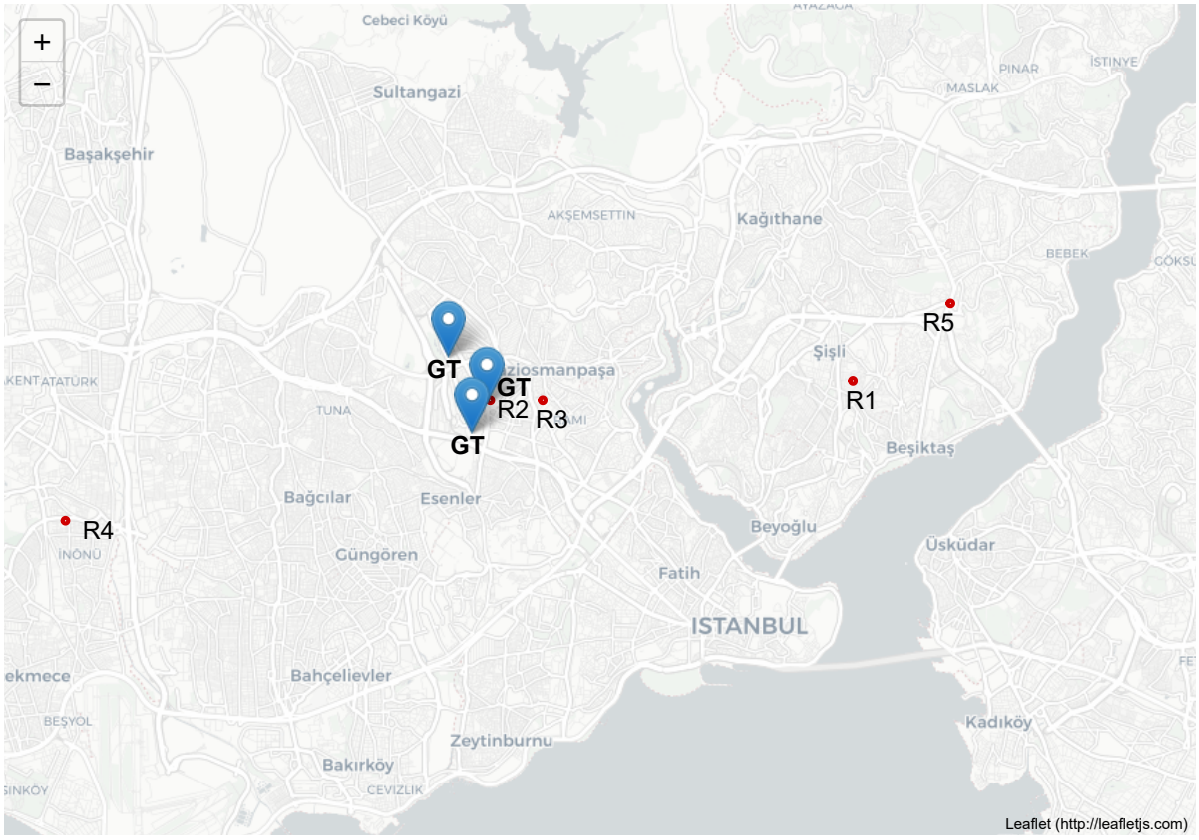

Figure 6.7 Map Visualization of alternative method missing recommendations for a sample user

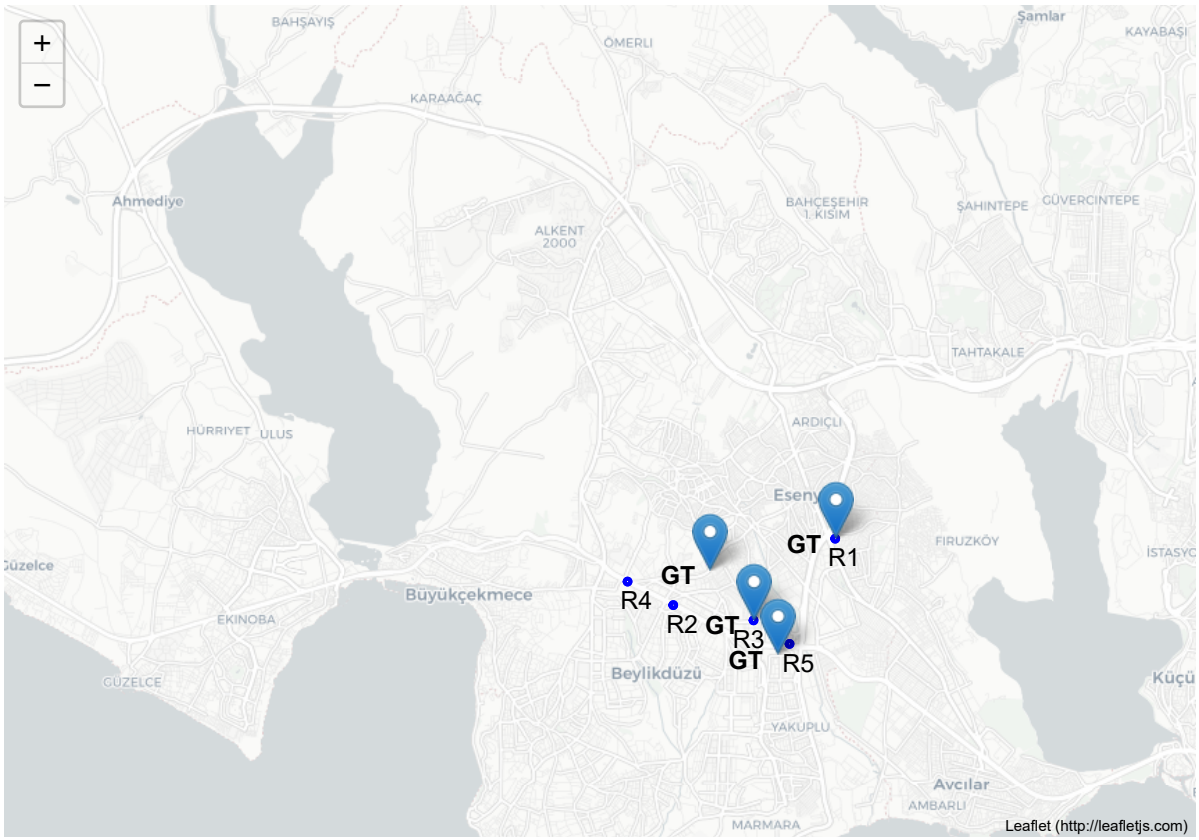

Figure 6.8 Map Visualization of proposed method catching recommendations for a sample user

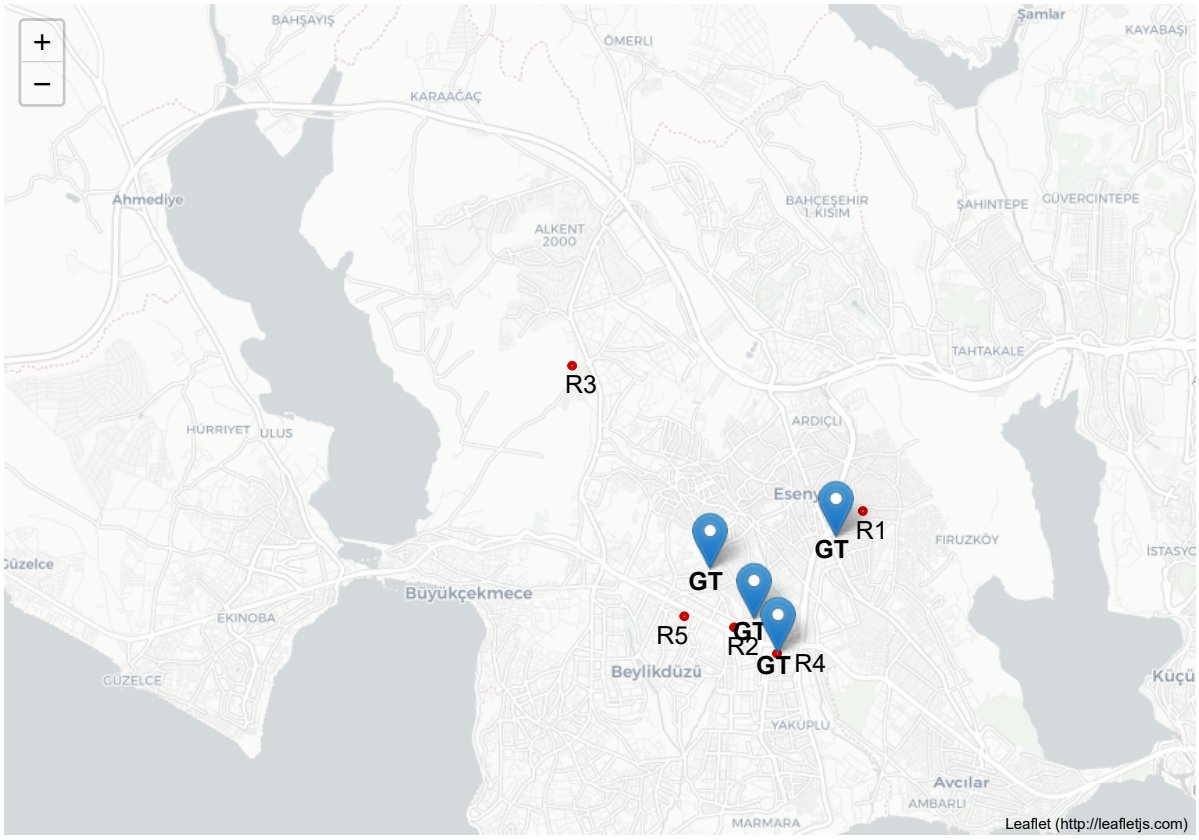

Figure 6.9 Map Visualization of alternative method catching recommendations for a sample user

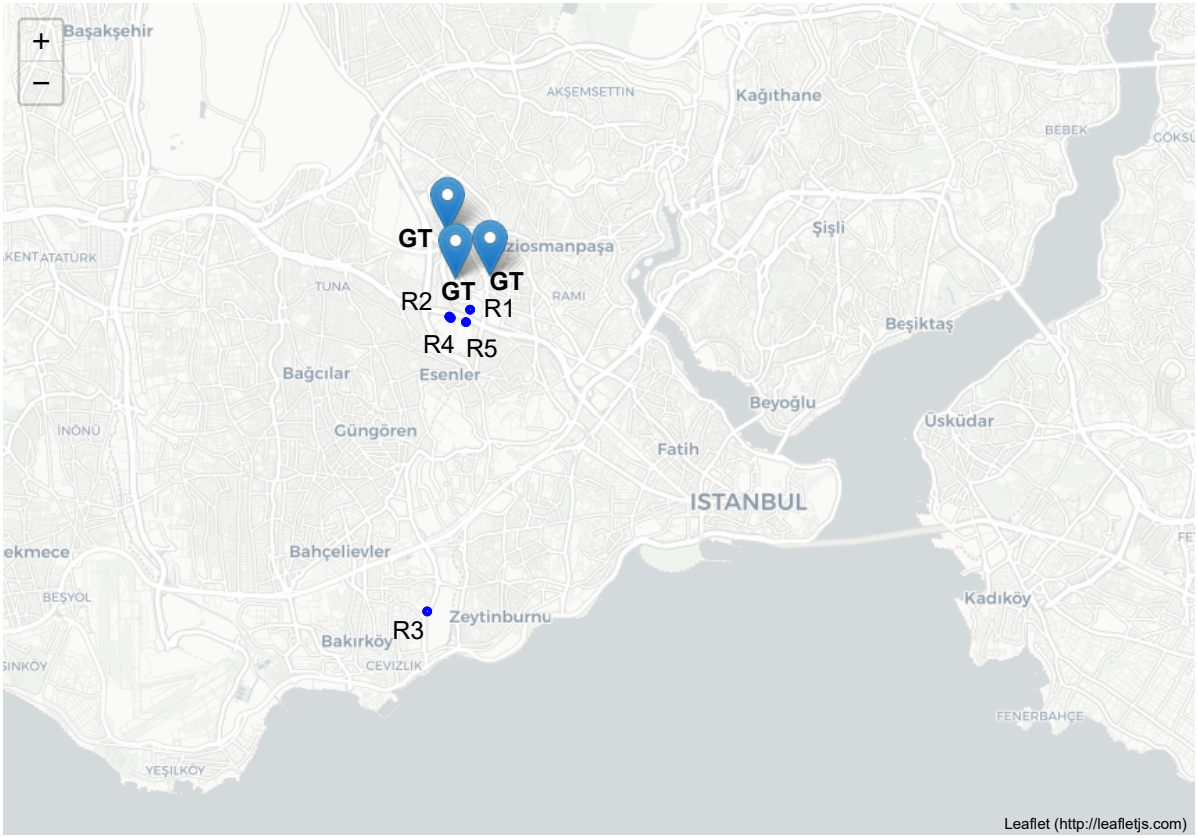

Figure 6.10 Map Visualization of proposed method missing recommendations for a sample user

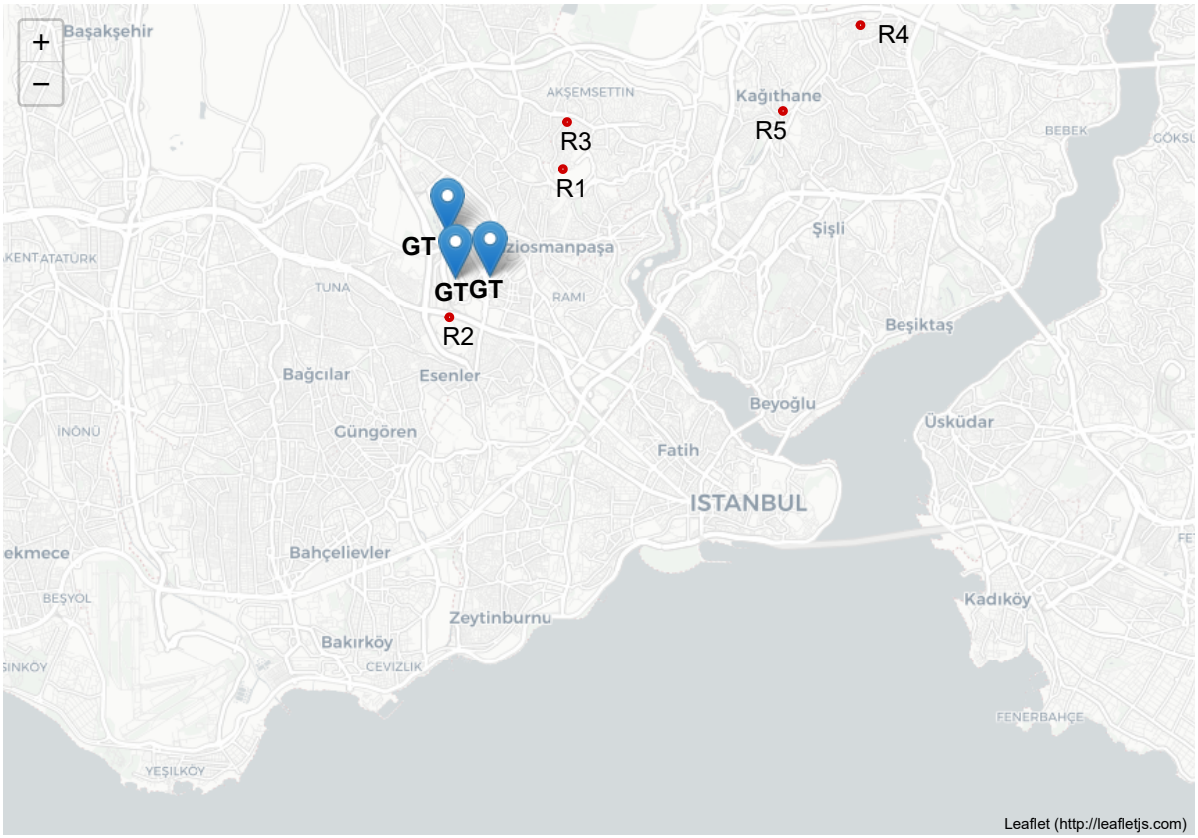

Figure 6.11 Map Visualization of alternative method missing recommendations for a sample user

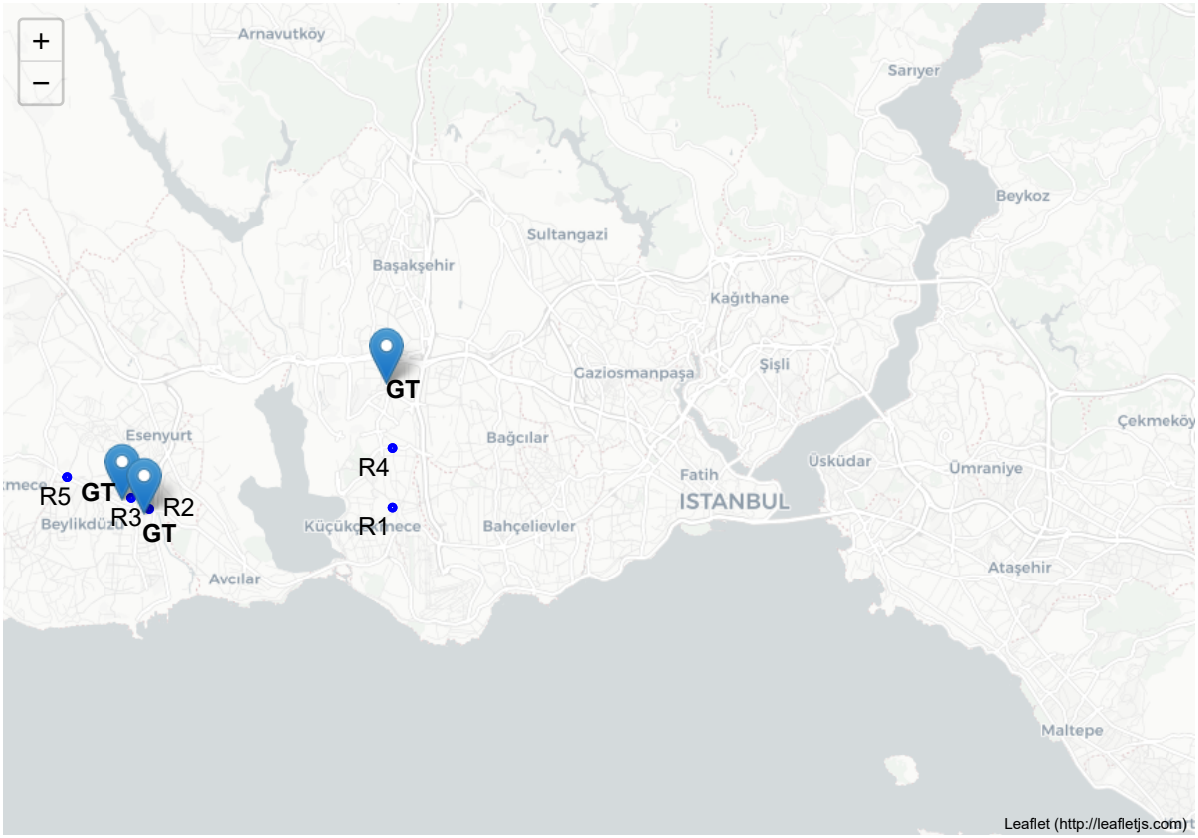

Figure 6.12 Map Visualization of proposed method missing recommendations for a sample user

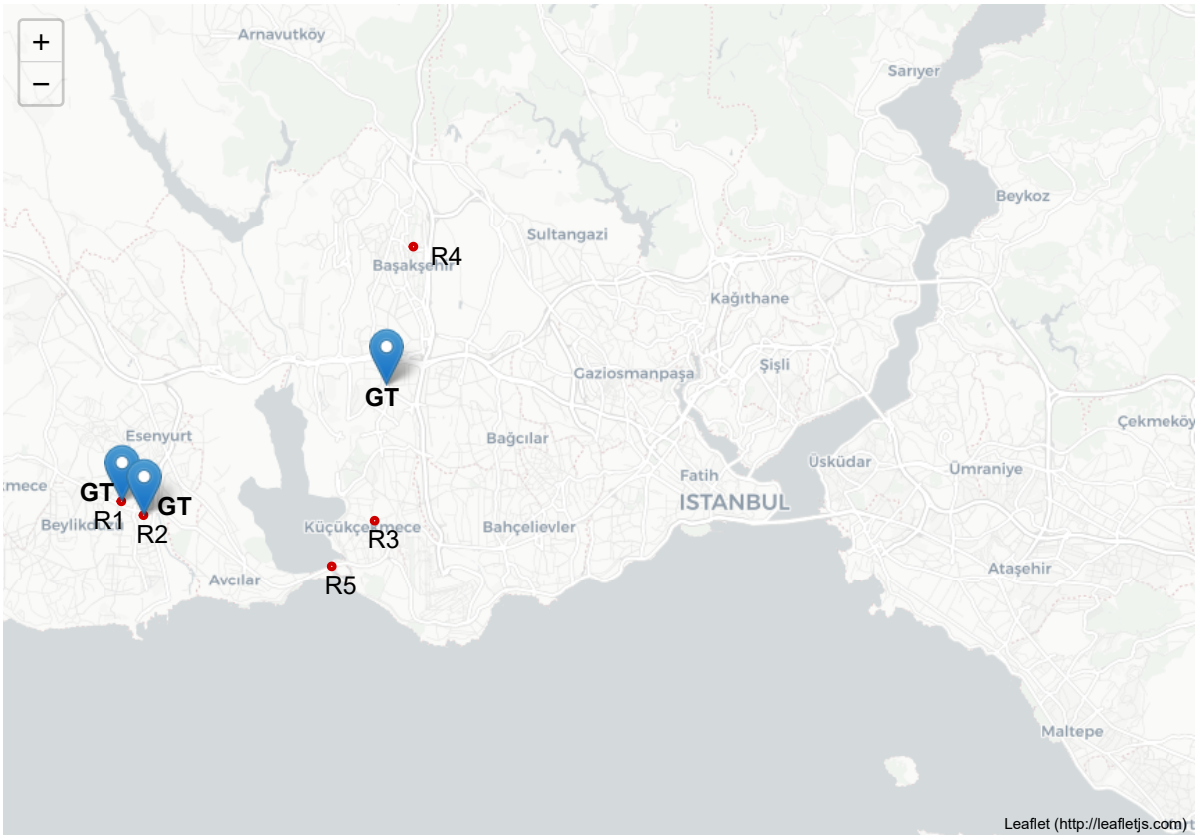

Figure 6.13 Map Visualization of alternative method catching recommendations for a sample user

Chapter 7

Conclusion and Future Work

In this thesis, a link prediction framework is proposed to create a scalable recommendation system for transaction data sets. The main goal of this thesis is to predict the most relevant future items where the user will establish a connection and make potential recommendations by use of proposed framework.

For this purpose, a link prediction based framework is proposed. It is using a binary classification approach where actual links between the pair of nodes in the network are classified as "positive" and the links that do not exist are classified as "negative".

Summary of Work

Proposed framework includes a preprocessing, embedding model and classifier model. At preprocessing step, transaction data set is put into a form of a network. By use of embedding methods, a latent space representation of the network is extracted to be used in a machine learning task. Binary operators are used to create forms of link embeddings from node representations. A classifier model is trained with the created link embeddings and used to classify test links in a binary classification framework that creates recommendations.

Several types of embedding and classifier models are examined, and their performances are analyzed on a use case scenario which is a merchant prediction task in a transaction data set. metapath2vec and node2vec are the embedding models that are used in this thesis. Gradient Boosting Decision Tree algorithms such as LightGBM, XGBoost, and another type of classifier Artifical Neural Networks are used as classifier models. Moreover, binary operators considered in this thesis are averaging, hadamard and l2.

Experiments are conducted in order to select the best embedding, classifier and binary operator combination. According to validation results evaluated on ROC, AUC and MAP@K metrics; the best combination is found to be metapath2vec, LightGBM, and average operator.

After model and operator selection, proposed model is compared with a collaborative filtering based alternative method, namely Alternating Least Squares. According to comparative analysis, proposed method shows superior performance than alternative method in terms of MAP@K, AUC scores, and detection rates in top-5 merchant recommendations. Also, a significance test is performed, and proposed method is found to have higher U-Test statistics in comparison with alternative method.

According to detection rate results, proposed method has caught more merchant purchases compared to alternative method as in their top-5 recommendations. It also means that even both models are catching the same merchant purchases, proposed method catches these merchants in the higher-ranked recommendations in comparison with alternative method.

A map comparison is also conducted, which is geographically representing recommended merchants in a map, and according to these map visualizations, proposed method is found to be making closer merchant recommendations to ground truth merchants geographically, even it cannot detect ground truth merchants exactly. These results can also be useful to have location information about customers to make recommendations more relevant.

According to significance results, p-values of both methods have significant performances where can be understood by their p-values are in the range smaller than 0.05. Since p-value distribution of proposed method is in smaller ranges, and its U-Test statistics are higher than alternative method, it can be concluded that proposed method has more significant results. Positive link prediction scores are significantly higher than negative prediction scores in proposed method in comparison with alternative method.

Overall, proposed method demonstrates a convincing performance that can be used as a recommendation system. The items which have the highest prediction scores can be utilized as top recommendations of a recommendation system. Moreover, it shows that using only previous interaction information can be a powerful signal that does not need another user or item features. Furthermore, due to the generalization power of proposed method, it can be used in many different types of data sets even they do not have a transaction nature.

Future Work

As future work, due to the existence of adjustable stages and high applicability of proposed method, it can open a new path for further research. Each stage in proposed method can be revisited and contributed from several perspectives.

Embedding models in this thesis adopt a random walk approach and generates the walks based on preserving neighborhood probabilities. Heterogeneous graphs use metapath2vec, and the homogeneous ones use node2vec widely. This process can be combined with a sub-graph extraction stage which later can be used for generalization of the network, and random walks can be generated based on these sub-graphs instead of using more hyper-parameters. Moreover, other similarity measures combined with cosine similarity can also be introduced as a further approach in embedding extraction.

Time dimension of the transactions can be benefited, which has not been used in this thesis. This information may be helpful in assigning weights to the links. Furthermore, the number of occurrence times of user-item pairs can also be used as weighted combination with time information to assign link weights. Binary operators can be revisited, or different link formation methods other than element-wise operators can be adopted. A dimensionality reduction stage can also be integrated in order to decrease training time and reduce overfitting. Use case scenarios can be expanded with product recommendations, song or movie recommendations based on the previous purchases, listening or watching history data in order to perform more performance analysis of the proposed method.

In addition, decreasing the number of pairwise combinations can be considered to have a more scalable system. It can be achieved by using node features. For example, a subset of merchants can be taken into consideration according to a location filter for a customer. This location information can also be understood; even they are not present explicitly as a customer node feature. Since proposed method makes some recommendations in some neighborhoods, this information can be inferred from these top recommendations and can be used to make new recommendations in these neighborhoods that decrease the number of pairwise combinations.

Likewise many other link prediction techniques, proposed method only operates on the "seen" nodes. It actually tries to discover the hidden links among the observed network. Unseen users and items can be taken into consideration by estimating their embeddings based on their external features which later can be used to test link existence in proposed method. This would be a solution to a common "cold start" problem in recommendation systems.

Bibliography

[1] Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 855–864, 2016.

[2] Jure Leskovec. Node embeddings. *CS224W: Machine Learning with Graphs lecture notes*, pages 9–10, 07 2021. Stanford University.

[3] David Easley and Jon Kleinberg. *Networks, Crowds, and Markets: Reasoning about a Highly Connected World*. Cambridge University Press, 2010.

[4] David Liben-Nowell and Jon Kleinberg. The link-prediction problem for social networks. *Journal of the American Society for Information Science and Technology*, 58(7):1019–1031, 2007.

[5] Albert-Laszlo Barabasi and Zoltan Oltvai. Network biology: Understanding the cell's functional organization. *Nature reviews. Genetics*, 5:101–13, 03 2004.

[6] Alex Fout, Jonathon Byrd, Basir Shariat, and Asa Ben-Hur. Protein interface prediction using graph convolutional networks. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 6533–6542, Red Hook, NY, USA, 2017. Curran Associates Inc.

[7] Peter W. Battaglia, Razvan Pascanu, Matthew Lai, Danilo Rezende, and Koray Kavukcuoglu. Interaction networks for learning about objects, relations and physics, 2016.

[8] Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. Dbpedia: A nucleus for a web of open data. In Karl Aberer, Key-Sun Choi, Natasha Noy, Dean Allemang, Kyung-Il Lee, Lyndon Nixon, Jennifer Golbeck, Peter Mika, Diana Maynard, Riichiro Mizoguchi, Guus Schreiber, and Philippe Cudré-Mauroux, editors, *The Semantic Web*, pages 722–735, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.

[9] Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. Open graph benchmark: Datasets for machine learning on graphs, 2021.

[10] Hsinchun Chen, Xin Li, and Zan Huang. Link prediction approach to collaborative filtering. In *Proceedings of the 5th ACM/IEEE-CS Joint Conference on Digital Libraries (JCDL '05)*, pages 141–142, 2005.

[11] Yuxiao Dong, Nitesh V Chawla, and Ananthram Swami. metapath2vec: Scalable representation learning for heterogeneous networks. In *KDD '17*, pages 135–144. ACM, 2017.

[12] Tianqi Chen and Carlos Guestrin. XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, pages 785–794, New York, NY, USA, 2016. ACM.

[13] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. Lightgbm: A highly efficient gradient boosting decision tree. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[14] Yifan Hu, Yehuda Koren, and Chris Volinsky. Collaborative filtering for implicit feedback datasets. In *2008 Eighth IEEE International Conference on Data Mining*, pages 263–272, 2008.

[15] Md Islam, Sabeur Aridhi, and Malika Smail. A comparative study of similarity-based and gnn-based link prediction approaches. 08 2020.

[16] Dekang Lin. An information-theoretic definition of similarity. In *Proceedings of the Fifteenth International Conference on Machine Learning*, ICML '98, page 296–304, San Francisco, CA, USA, 1998. Morgan Kaufmann Publishers Inc.

[17] Peng Wang, Baowen Xu, Yurong Wu, and Xiaoyu Zhou. Link prediction in social networks: the state-of-the-art, 2014.

[18] M. E. J. Newman. Clustering and preferential attachment in growing networks. *Phys. Rev. E*, 64:025102, Jul 2001.

[19] Gerard Salton and Michael McGill. *Introduction to modern information retrieval*. McGraw-Hill, New York, NY, 1983.

[20] T. A. Sorensen. A method of establishing groups of equal amplitude in plant sociology based on similarity of species content and its application to analyses of the vegetation on danish commons. *Biol. Skar.*, 5:1–34, 1948.

[21] Paul Jaccard. Etude de la distribution florale dans une portion des alpes et du jura. *Bulletin de la Societe Vaudoise des Sciences Naturelles*, 37:547–579, 01 1901.

[22] Lada Adamic and Eytan Adar. Friends and neighbors on the web. *Social Networks*, 25:211–230, 2001.

[23] Leo Katz. A new status index derived from sociometric analysis. *Psychometrika*, 18(1):39–43, March 1953.

[24] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The pagerank citation ranking: Bringing order to the web. Technical Report 1999-66, November 1999.

[25] Edmund Landau. Zur relativen wertbemessung der turnierresultate deutsches wochenschach. *Jahrgang*, pages 366–369, 1895.

[26] Linyuan Lü, Ci-Hang Jin, and Tao Zhou. Similarity index based on local paths for link prediction of complex networks. *Phys. Rev. E*, 80:046122, Oct 2009.

[27] Linyuan Lü and Tao Zhou. Link prediction in complex networks: A survey. *Physica A: Statistical Mechanics and its Applications*, 390(6):1150–1170, Mar 2011.

[28] Aaron Clauset, Cristopher Moore, and M. E. J. Newman. Hierarchical structure and the prediction of missing links in networks. *Nature*, 453(7191):98–101, May 2008.

[29] Harrison C. White, Scott A. Boorman, and Ronald L. Breiger. Social structure from multiple networks. i. blockmodels of roles and positions. *American Journal of Sociology*, 81(4):730–780, 1976.

[30] Paul W. Holland, Kathryn Blackmond Laskey, and Samuel Leinhardt. Stochastic blockmodels: First steps. *Social Networks*, 5(2):109–137, 1983.

[31] Clement Lee and Darren J. Wilkinson. A review of stochastic block models and extensions for graph clustering. *Applied Network Science*, 4(1), Dec 2019.

[32] Lise Getoor, Nir Friedman, Daphne Koller, Avi Pfeffer, and Ben Taskar. Probabilistic relational models, 2001.

[33] Lise Getoor and Lilyana Mihalkova. Learning statistical models from relational data. pages 1195–1198, 06 2011.

[34] Kai Yu, Wei Chu, Shipeng Yu, Volker Tresp, and Zhao Xu. Stochastic relational models for discriminative link prediction. In B. Schölkopf, J. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems*, volume 19. MIT Press, 2007.

[35] Muhan Zhang and Yixin Chen. Link prediction based on graph neural networks. In *Advances in Neural Information Processing Systems*, 2018.

[36] Muhan Zhang and Yixin Chen. Weisfeiler-lehman neural machine for link prediction. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '17, page 575–583, New York, NY, USA, 2017. Association for Computing Machinery.

[37] William Hamilton, Rex Ying, and Jure Leskovec. Representation learning on graphs: Methods and applications. 09 2017.

[38] Brian Gallagher and Tina Eliassi-Rad. Leveraging label-independent features for classification in sparsely labeled networks: An empirical study. In *Proceedings of the Second International Conference on Advances in Social Network Mining and Analysis*, SNAKDD'08, page 1–19, Berlin, Heidelberg, 2008. Springer-Verlag.

[39] Keith Henderson, Brian Gallagher, Lei Li, Leman Akoglu, Tina Eliassi-Rad, Hanghang Tong, and Christos Faloutsos. It's who you know: Graph mining using recursive structural features. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '11, page 663–671, New York, NY, USA, 2011. Association for Computing Machinery.

[40] Mikhail Belkin and Partha Niyogi. Laplacian eigenmaps and spectral techniques for embedding and clustering. In *Proceedings of the 14th International Conference on Neural Information Processing Systems: Natural and Synthetic*, NIPS'01, page 585–591, Cambridge, MA, USA, 2001. MIT Press.

[41] I.T. Jolliffe. *Principal Component Analysis*. Springer Verlag, 1986.

[42] Joshua B. Tenenbaum, Vin de Silva, and John C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323, 2000.

[43] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(86):2579–2605, 2008.

[44] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space, 2013.

[45] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk. *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, Aug 2014.

[46] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. Line. *Proceedings of the 24th International Conference on World Wide Web*, May 2015.

[47] William L. Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs, 2017.

[48] M. Kearns and L. G. Valiant. Learning boolean formulae or finite automata is as hard as factoring. Technical Report TR 14-88, Harvard University Aiken Computation Laboratory, 1988.

[49] Michael Kearns and Leslie Valiant. Cryptographic limitations on learning boolean formulae and finite automata. *J. ACM*, 41(1):67–95, January 1994.

[50] Robert E. Schapire. A brief introduction to boosting. In *Proceedings of the 16th International Joint Conference on Artificial Intelligence - Volume 2*, IJCAI'99, page 1401–1406, San Francisco, CA, USA, 1999. Morgan Kaufmann Publishers Inc.

[51] Jerome H. Friedman. Greedy function approximation: A gradient boosting machine. *The Annals of Statistics*, 29(5):1189–1232, 2001.

[52] Michael J. Pazzani and Daniel Billsus. *Content-Based Recommendation Systems*, page 325–341. Springer-Verlag, Berlin, Heidelberg, 2007.

[53] H. P. Luhn. A statistical approach to mechanized encoding and searching of literary information. *IBM Journal of Research and Development*, 1(4):309–317, 1957.

[54] H. P. Luhn. A statistical approach to mechanized encoding and searching of literary information. *IBM Journal of Research and Development*, 1(4):309–317, 1957.

[55] Donghui Wang, Yanchun Liang, Dong Xu, Xiaoyue Feng, and Renchu Guan. A content-based recommender system for computer science publications. *Knowledge-Based Systems*, 157:1–9, 2018.

[56] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th International Conference on World Wide Web*, WWW '01, page 285–295, New York, NY, USA, 2001. Association for Computing Machinery.

[57] Zan Huang, Wingyan Chung, and Hsinchun Chen. A graph model for e-commerce recommender systems. *J. Am. Soc. Inf. Sci. Technol.*, 55(3):259–274, February 2004.

[58] Xiaoyuan Su and Taghi M. Khoshgoftaar. A survey of collaborative filtering techniques. *Adv. in Artif. Intell.*, 2009, January 2009.

[59] T. Jaya Lakshmi and S. Durga Bhavani. Link prediction approach to recommender systems. *CoRR*, abs/2102.09185, 2021.

[60] Nathan Srebro, Jason Rennie, and Tommi Jaakkola. Maximum-margin matrix factorization. In L. Saul, Y. Weiss, and L. Bottou, editors, *Advances in Neural Information Processing Systems*, volume 17. MIT Press, 2005.

[61] Jasson D. M. Rennie and Nathan Srebro. Fast maximum margin matrix factorization for collaborative prediction. In *Proceedings of the 22nd International Conference on Machine Learning*, ICML '05, page 713–719, New York, NY, USA, 2005. Association for Computing Machinery.

[62] Dennis DeCoste. Collaborative prediction using ensembles of maximum margin matrix factorizations. In *Proceedings of the 23rd International Conference on Machine Learning*, ICML '06, page 249–256, New York, NY, USA, 2006. Association for Computing Machinery.

[63] Robert Bell, Yehuda Koren, and Chris Volinsky. Modeling relationships at multiple scales to improve accuracy of large recommender systems. In *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '07, page 95–104, New York, NY, USA, 2007. Association for Computing Machinery.

[64] Daniel Billsus and Michael J. Pazzani. Learning collaborative information filters. In *Proceedings of the Fifteenth International Conference on Machine Learning*, ICML '98, page 46–54, San Francisco, CA, USA, 1998. Morgan Kaufmann Publishers Inc.

[65] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, 2009.

[66] Yunhong Zhou, Dennis Wilkinson, Robert Schreiber, and Rong Pan. Large-scale parallel collaborative filtering for the netflix prize. In *Proc. 4th Int'l Conf. Algorithmic Aspects in Information and Management, LNCS 5034*, pages 337–348. Springer, 2008.

[67] Tao Zhou, Jie Ren, Matú š Medo, and Yi-Cheng Zhang. Bipartite network projection and personal recommendation. *Phys. Rev. E*, 76:046115, Oct 2007.

[68] Zhijun Yin, Manish Gupta, Tim Weninger, and Jiawei Han. A unified framework for link recommendation using random walks. In *Proceedings of the 2010 International Conference on Advances in Social Networks Analysis and Mining*, ASONAM '10, page 152–159, USA, 2010. IEEE Computer Society.

[69] Feng Xie, Zhen Chen, Jiaxing Shang, Xiaoping Feng, and Jun Li. A link prediction approach for item recommendation with complex number. *Knowledge-Based Systems*, 81:148–158, 2015.

[70] Yuxiao Dong, Jie Tang, Sen Wu, Jilei Tian, Nitesh V. Chawla, Jinghai Rao, and Huanhuan Cao. Link prediction and recommendation across heterogeneous social networks. In *2012 IEEE 12th International Conference on Data Mining*, pages 181–190, 2012.

[71] Nitin Chiluka, Nazareno Andrade, and J.A. Pouwelse. A link prediction approach to recommendations in large-scale user-generated content systems. pages 189–200, 04 2011.

[72] Zhiyuan He, Danchen Lin, Thomas Lau, and Mike Wu. Gradient boosting machine: A survey. 08 2019.

[73] Nigel Duffy and David Helmbold. A geometric approach to leveraging weak learners. In Paul Fischer and Hans Ulrich Simon, editors, *Computational Learning Theory*, pages 18–33, Berlin, Heidelberg, 1999. Springer Berlin Heidelberg.

[74] Tianqi Chen. Introduction to boosted trees. *CS 675: Introduction to Machine learning lecture notes*, 2014. Washington University.

[75] Guido Van Rossum and Fred L Drake Jr. *Python tutorial*. Centrum voor Wiskunde en Informatica Amsterdam, The Netherlands, 1995.

[76] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015.

[77] Peter Fader and Bruce Hardie. Probability models for customer-base analysis. *Journal of Interactive Marketing - J INTERACT MARK*, 23:61–69, 02 2009.

[78] Saar Kagan and Ron Bekkerman. Predicting purchase behavior of website audiences. *International Journal of Electronic Commerce*, 22(4):510–539, 2018.

[79] A machine learning framework for customer purchase prediction in the non-contractual setting. *European Journal of Operational Research*, 281(3):588–596, 2020. Featured Cluster: Business Analytics: Defining the field and identifying a research agenda.

[80] Jon Goss. "we know who you are and we know where you live": The instrumental rationality of geodemographic systems. *Economic Geography*, 71(2):171–198, 1995.

[81] Charlotte Mason. *Journal of Marketing Research*, 39(4):499–501, 2002.

[82] H. B. Mann and D. R. Whitney. On a test of whether one of two random variables is stochastically larger than the other. *The Annals of Mathematical Statistics*, 18(1):50–60, 1947.

APPENDIX A
Node Embedding Visualization

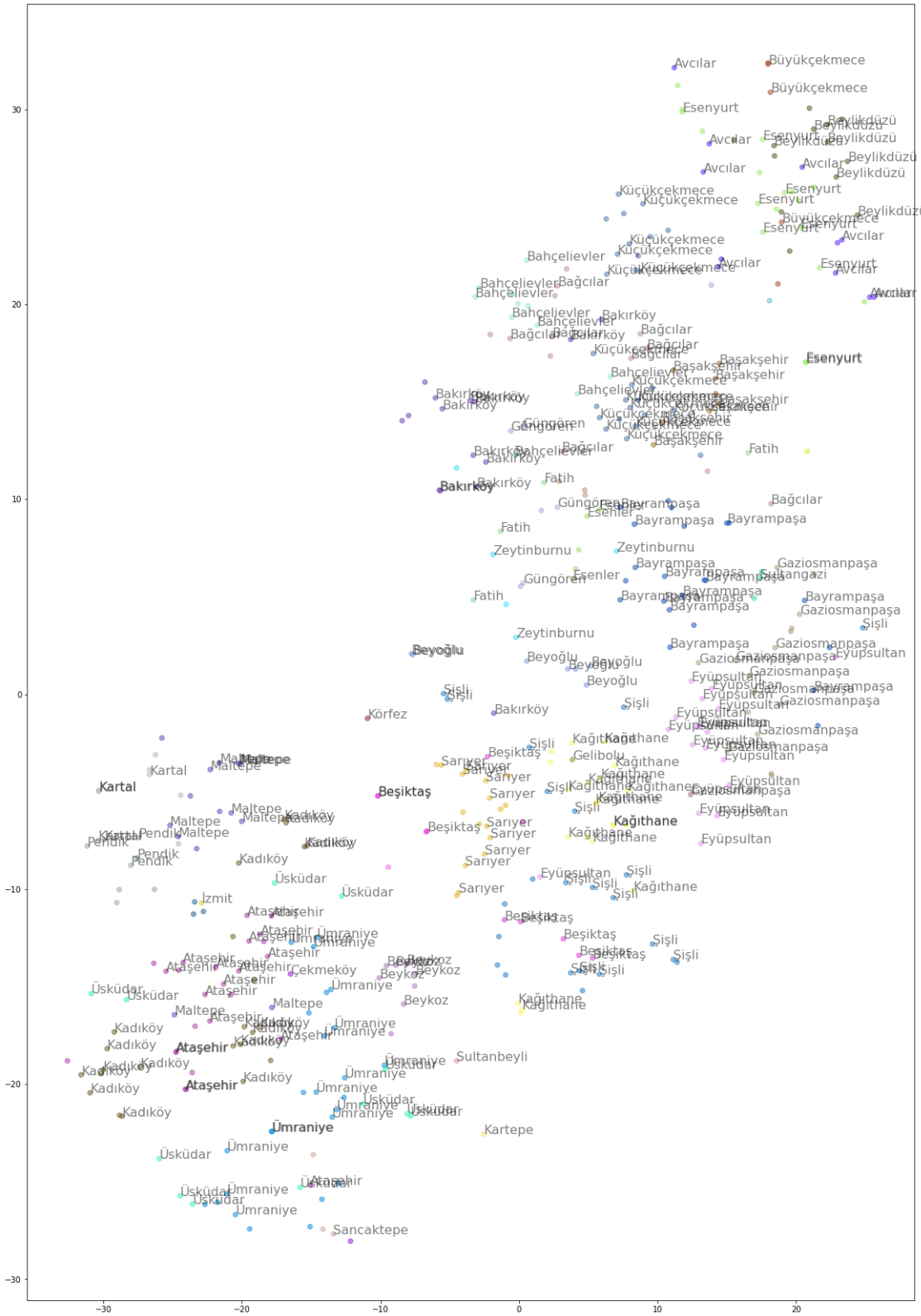

Figure A.1 Town labeled metapath2vec merchant node embedding visualization

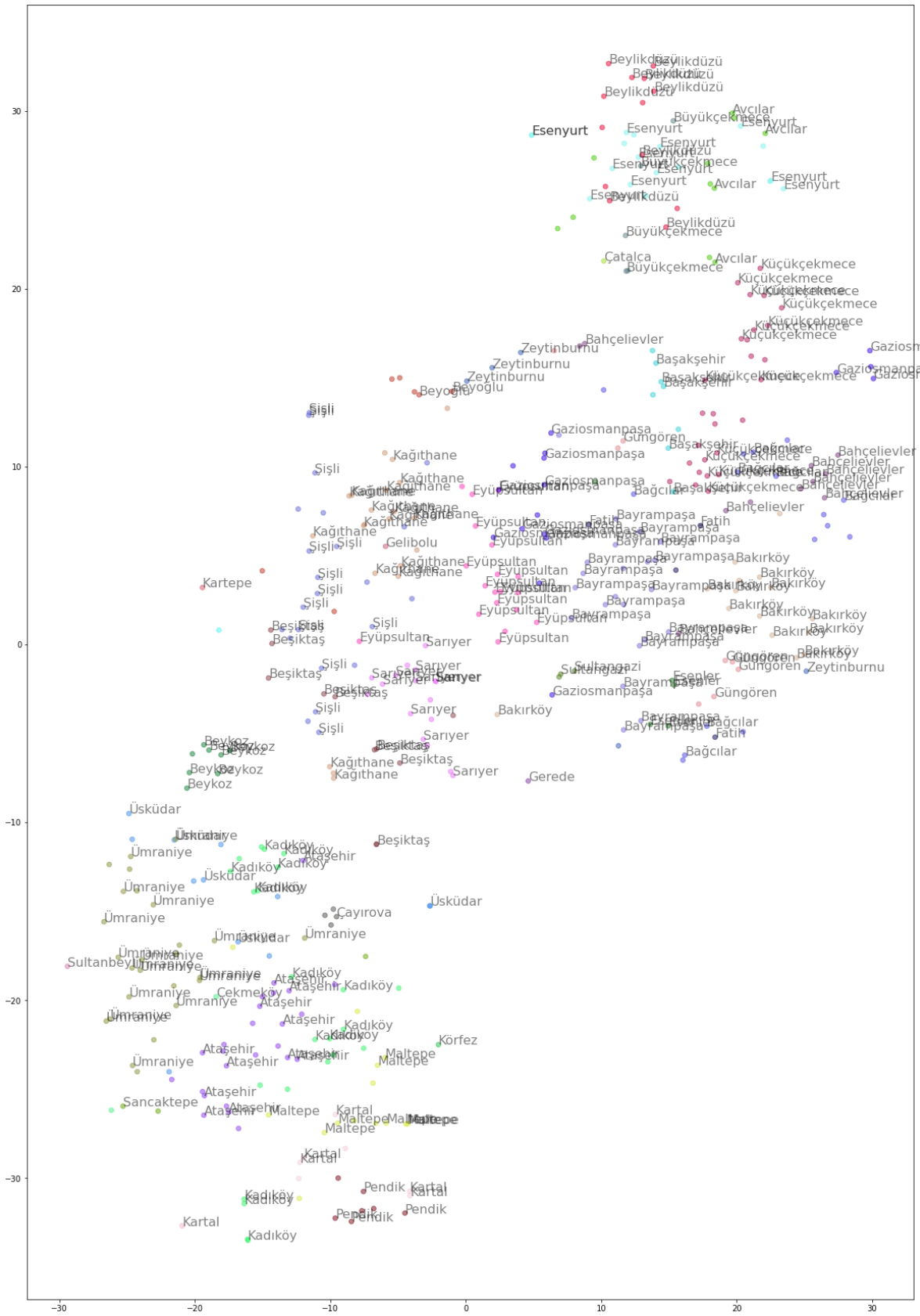

Figure A.2 Town labeled node2vec merchant node embedding visualization

Link Embedding Visualization

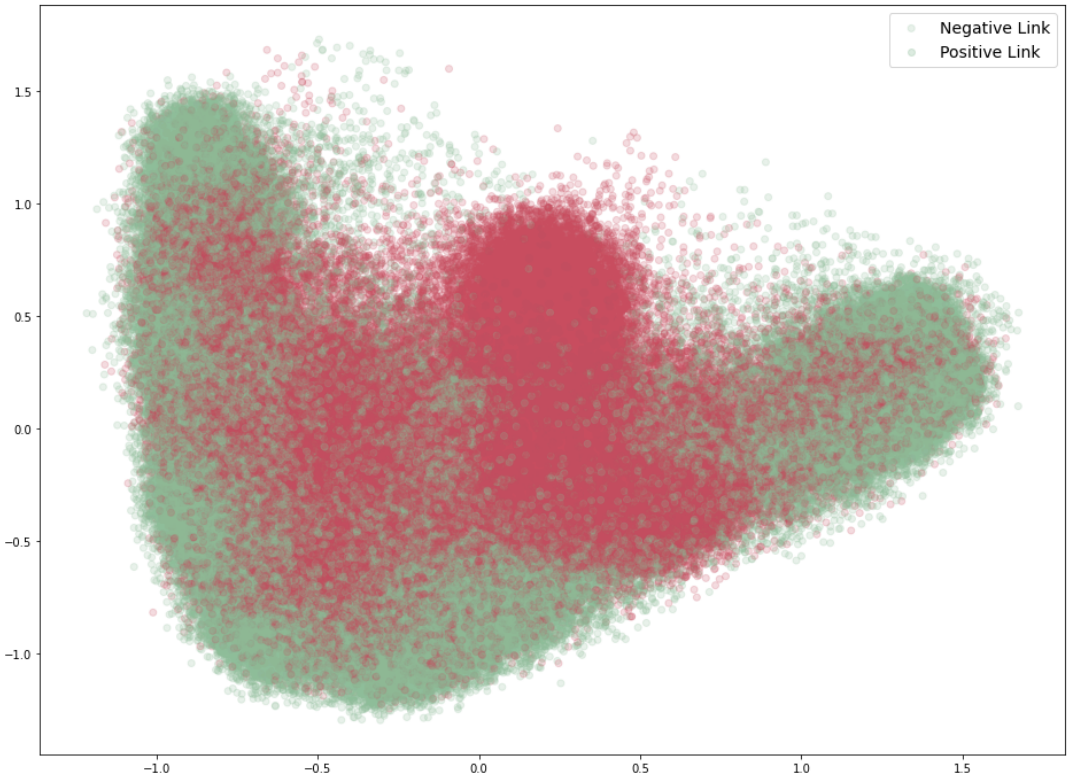

Figure A.3 Metapath2vec Sampled Train Link Embedding Visualization

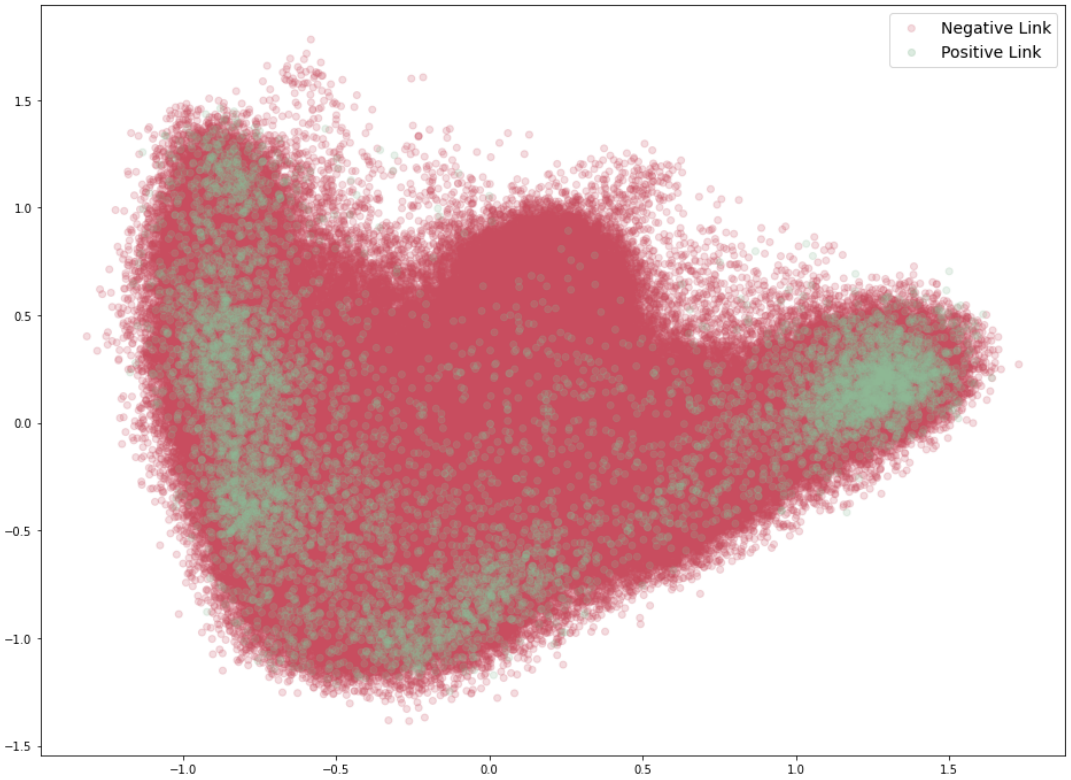

Figure A.4 Node2vec Sampled Test Link Embedding Visualization

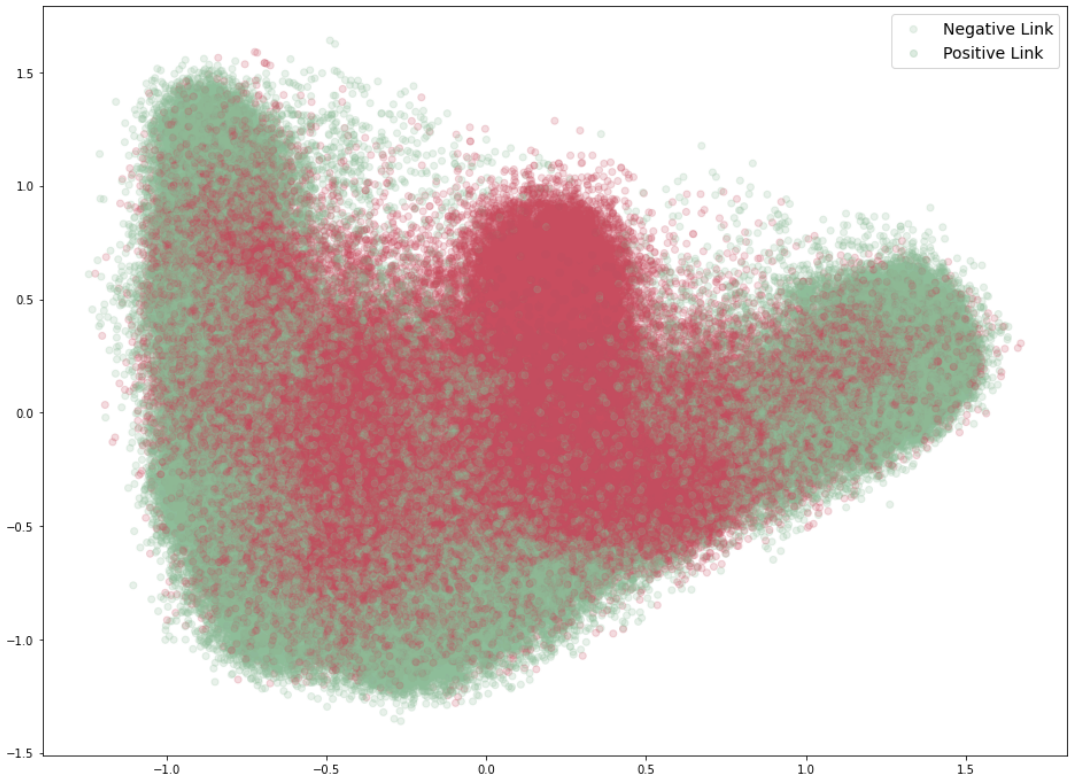

Figure A.5 Node2vec Sampled Train Link Embedding Visualization

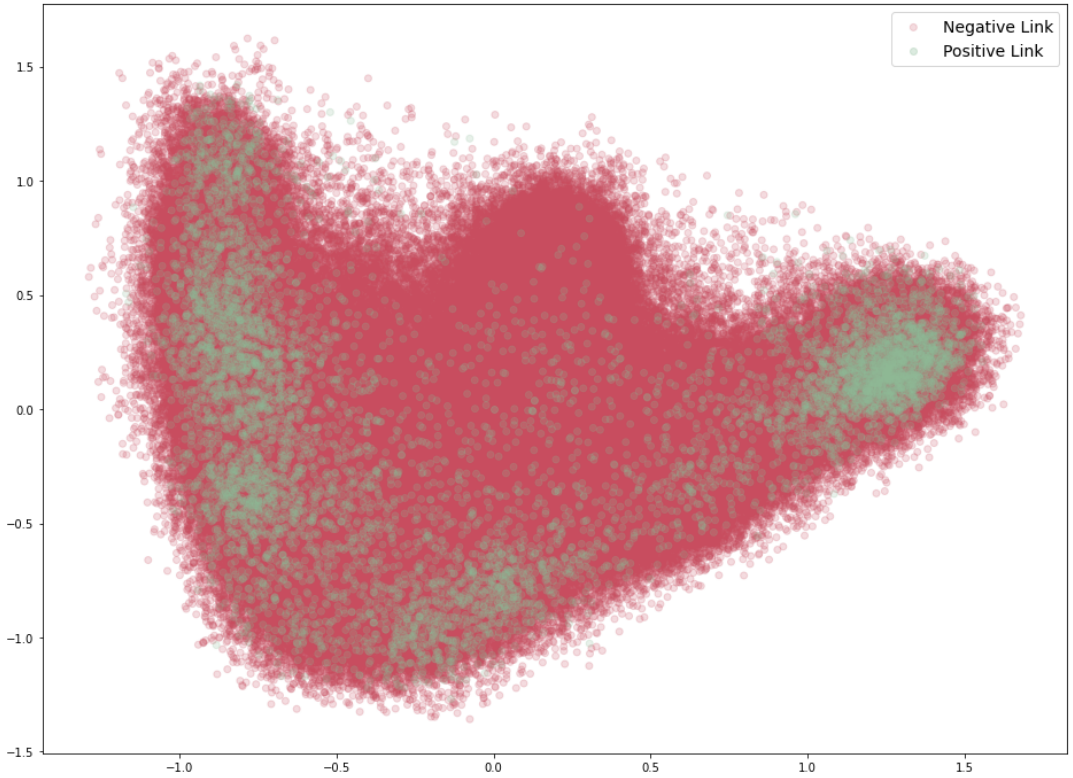

Figure A.6 Node2vec Sampled Test Link Embedding Visualization