**ISTANBUL TECHNICAL UNIVERSITY ★ GRADUATE SCHOOL**

# INVESTIGATION OF EMBEDDING METHODS ON GNN BASED RECOMMENDER MODELS

**M.Sc. THESIS**

**Mustafa AKER**

**Department of Computer Engineering**

**Computer Engineering Programme**

**JULY 2021**

**ISTANBUL TECHNICAL UNIVERSITY ★ GRADUATE SCHOOL**

# INVESTIGATION OF EMBEDDING METHODS ON GNN BASED RECOMMENDER MODELS

**M.Sc. THESIS**

**Mustafa AKER**
**504171553**

**Department of Computer Engineering**

**Computer Engineering Programme**

**Thesis Advisor: Asst. Prof. Dr. Yusuf Yaslan**

**JULY 2021**

**ISTANBUL TEKNİK ÜNİVERSİTESİ ★ LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ**

# ÇİZGE SİNİR AĞI TEMELLİ ÖNERİ SİSTEMLERİ ÜZERİNDE KULLANILAN GÖMME TİPİNİN ETKİSİNİN ARAŞTIRILMASI

**YÜKSEK LİSANS TEZİ**

**Mustafa AKER**
**504171553**

**Bilgisayar Mühendisliği Anabilim Dalı**

**Bilgisayar Mühendisliği Programı**

**Tez Danışmanı: Dr. Öğr. Üyesi Yusuf Yaslan**

**TEMMUZ 2021**

Mustafa Aker, a M.Sc. student of ITU Graduate School student ID 504171553 successfully defended the thesis/dissertation entitled “Investigation of Embedding Methods on GNN Based Recommender Models”, which he prepared after fulfilling the requirements specified in the associated legislations, before the jury whose signatures are below.

| | | |
|---|---|---|
| **Thesis Advisor :** | **Asst. Prof. Dr. Yusuf Yaslan**<br>Istanbul Technical University | ..... |
| **Jury Members :** | **Assoc. Prof. Dr. Ahmet Cüneyd Tantuğ**<br>Istanbul Technical University | ..... |
| | **Asst. Prof. Dr. Pınar Sarısaray Bölük**<br>Bahcesehir University | ..... |

**Date of Submission : 11.06.2021**
**Date of Defense :**

***To my family and dearest friends,***

**FOREWORD**

Thanks to everyone who contributes any piece of moral and material support and made this work possible.

June 2021 Mustafa AKER

**TABLE OF CONTENTS**

**Page**

## ABBREVIATIONS

**AE** : Auto Encoder
**BPR** : Bayesian Personalized Ranking
**BSF** : Binary Step Function
**CBF** : Content-Based Filtering
**CBOW** : Common Bag of Words
**CF** : Collaborative Filtering
**CNN** : Convolutional Neural Network
**CPU** : Central Processing Unit
**DNGR** : Deep Neural Networks for Learning Graph Representations
**GAE** : Graph Auto Encoder
**GAT** : Graph Attention Network
**GCN** : Graph Convolutional Network
**GFF** : General Factorization Framework
**GGNN** : Gated Graph Neural Network
**GHz** : Giga Hertz
**GNN** : Graph Neural Network
**GPU** : Graphics Processing Unit
**GraphGAN** : Graph Representation Learning with Generative Adversarial Nets
**HOPE** : High Order Proximity Preserved Embedding
**Knn** : K Nearest Neighbor
**LINE** : Large-scale Information Network Embedding
**MB** : Mega Bytes
**MDP** : Markov Decision Process
**MLP** : Multi Layer Perceptron
**NARM** : Neural Attentive Recommendation Machine
**RAM** : Random Access Memory
**ReLU** : Rectified Linear Unit
**RNN** : Recurrent Neural Network
**RS** : Recommender System
**SBRS** : Session Based Recommender System

**SDNE** **:** Structural Deep Network Embedding

**SR-GNN** **:** Session-Based Recommendation with Graph Neural Networks

**STAMP** **:** Short-Term Attention/Memory Priority Model

**STMP** **:** Short-Term Memory Priority

**VGAE** **:** Variational Graph Auto Encoder

**VRAM** **:** Video Random Access Memory

## LIST OF TABLES

**Page**

**LIST OF FIGURES**

**Page**

## INVESTIGATION OF EMBEDDING METHODS ON GNN BASED RECOMMENDER MODELS

## SUMMARY

Digitalization of the tasks that touch the daily life of human beings, became one of the major processes during the 21st century. This sight forces the processes that require physical human interaction to handle by computerized systems. Ranging from marketing applications to healthcare systems, this digitalization era is affecting various sections of human daily life. As a result of the huge developments in hardware and software components, computerized systems were allowed to grow at an unseen pace. Most of the companies that do not want to miss the chance of the downwind effect of this digital transformation, adapted their systems in order to keep pace with this rush. From the perspective of this evolution, e-marketing and e-commerce fields are the ones that lie in the heart of this digitalization process.

Along with the digital transformation, a huge amount of data started to be collected from their computerized infrastructures. Therefore, it becomes viable to obtain and use the information that is collected from the cumulative data. Being aware of which data is beneficial for the organization is considered to be one of the most critical key points. These data collection and data processing will unlock the way of knowing their customer base in a better way in terms of their features, behaviors, and relationships. However, only collecting data alone is not sufficient for having advanced systems that address to draw the attention of possible new customers. Power of the collected data is unleashed when it is combined with an intelligent processing system.

Considering processing and gathering a huge amount of data, the problem comes into the domain of machine learning and deep learning fields. Thanks to the recent developments in the field of computer hardware systems such as introducing new CPU and GPU architectures and also, computer software systems like publishing new powerful and efficient machine learning and deep learning libraries led those intelligent systems to become very popular in e-commerce and e-marketing fields.

Accordingly, the next item prediction specific to recommendation systems shows up as a challenging but rewarding task in related fields. Ability to learn the behaviors and preferences of the specific customer, become the supreme priority for those organizations to perform the best recommendations for the specific user. This will allow those organizations to be one step ahead compared to their competitors. Within the context of the session-based recommendation models, there exists no profiling operation that allows binding the features of sessions to the specific user which means that each session is treated individually. That makes the recommendation problem more challenging, due to the absence of sufficient data. At this point, researchers try to introduce a new breath to the problem by bringing a new perspective to the problem. Benefitting from graph structures during the representation of the sessions offers to use new features to the current solution by revealing session-specific relations.

As an application, several session-based item recommendation models were introduced by the researchers. Evolution of these models draws attention with the introduction of GRU4Rec and continues with NARM and STAMP session-based recommendation models. Lastly, publication of the SR-GNN method introduced a new state-of-art model in this field. By using these methodologies, it is possible to predict users' next behavior and recommend items according to their actions in a more efficient way.

As a nature of the session-based recommendation problem, there exists only the sequence information among the items. Therefore, representing the sessions in a more compact and more meaningful way has become the new research area. At this point, graph structures come to help. Representing the sessions via graph structures boosted the performance of the current recommendation models. So, investigation of SR-GNN model, which is the recommendation model based on graph representation, possibly reveals new research questions that will be beneficial for improving the overall model.

This research investigates the internal structure of the state-of-art graph neural network-based recommender model, SR-GNN, and expresses the test results by obtained making modifications to its internal modules. Accordingly, SR-GNN session-based recommendation model consists of three major modules which are pre-processing module, embedding module, and prediction module. Within the scope of the research, experiments conducted on SR-GNN model, research question mainly focused on the embedding module of the overall structure.

The embedding part of the SR-GNN recommendation model consists of two sub-modules which are item embedding sub-module and session embedding sub-module. Sticking with the graph embedding scheme, all sessions that last inside the dataset, represented as graph structures where items are assigned to the nodes and relationships among the items assigned to the edges that connect these nodes. Then, adjacency matrices of each session are given to the embedding module of the model to obtain their vector representations. In the session embedding module of the vanilla model, embeddings of the sessions are calculated by averaging their normalized weights of the edges. After completing specific embedding calculations in sub-modules, obtained results are merged together and a matrix that represents the Euclidean distance among each item embeddings is given as an input to the prediction module of the overall structure.

Within the scope of this study session embedding sub-module of the GNN based recommender model has been modified by applying different embedding techniques which are singular value decomposition, Laplacian eigenmaps, word2vec, node2vec, Deepwalk, and LINE. During the experiments, two different datasets which are YOOCHOOSE and DIGINETICA have been used after executing specific pre-processing steps. Totally 12 different experiments have been conducted by applying introduced 6 different embedding methods on each dataset and performance results of these results have been collected. As evaluation metrics, Recall@20 and MRR@20 criteria which are widely used in the field of session-based recommendation models have been selected.

According to the obtained results, it is clearly seen that the embedding method used during the generation of session embeddings, affects the performance of the GNN based recommender model in different ways. The decision of which embedding technique should be selected for GNN based recommender model depends on the

internal structure of the dataset. Therefore, case-specific implementations seem to be the best solution to overcome the problem of selecting the embedding method.

# ÇİZGE SİNİR AĞI TEMELLİ TAVSİYE SİSTEMLERİ ÜZERİNDE KULLANILAN GÖMME TİPİNİN ETKİSİNİN ARAŞTIRILMASI

## ÖZET

Değişen ve dijitalleşen dünyada, insan yaşamına dokunan işlemlerin de dijitalleşmesi 21. yüzyılın başından itibaren önem kazanmıştır. Bu değişim kapsamında, insanların tarafından yüksek oranda fiziksel etkileşimin gerektiği işlemler bile, hızla bilgisayar destekli sistemler tarafından yürütülmektedir. E-ticaret uygulamalarından finansal işlemlerin yürütülmesine, sağlık sektöründe kullanılan sistemlerin geliştirilmesinden kişisel veya akademik eğitimlerin yürütülmesini kapsayan çok geniş bir yelpazeye hitap ederek, bu değişim süreci insanların günlük yaşantısının hemen hemen her alanında yer almaktadır. Buna ek olarak, yazılım ve donanım alanında görülen büyük gelişmeler bilgisayarlı sisteme geçiş sürecini öngörülemeyecek şekilde hızlandırmıştır. Çoğu organizasyonel yapı, bu geçiş sürecindeki rüzgârı arkasına alarak, halihazırda popüler bir seyir halinde olan bu akıma uyum sağlamak adına kendi sistemlerini adapte olmaya çalışmaktadırlar. Bu devrimsel dönüşüm bakış açısı göz önünde bulundurulduğunda, e-ticaret ve e-pazarlama alanları, bu değişin sürecinin tam merkezinde yer almaktadırlar.

Bilgisayarlı sistemlere geçiş ile birlikte çok yüksek miktarda verinin üretilmesi ve depolanması, dijitalleşme sürecinin bir sonucu olarak karşımıza çıkmaktadır. Bununla birlikte, depolanan veri miktarının artması sonucunda, biriken veriden faydalı bilgi elde etme çalışmaları da gündeme gelmeye başlamıştır. Bu noktada, eldeki verilerden faydalanarak, kuruluşlara yaralı olacak bilgiyi etme işi kritik bir öneme sahip olmaya başlamıştır. Bu bilgilerden faydalanmak, kurumlara kendi müşterilerini, özelliksel, davranışsal ve müşterilerinin birbirleri arasındaki ilişkilerini daha iyi tanıyabilme yolunda önemli ilerlemeler sağlamaktadır. Ancak, elde edilen veriyi kıymetlendirmeden yalnızca veri toplama işini gerçekleştirmek, yeni kullanıcıların ilgilisini çekme konusunda faydalı olmayacağından, firmaya herhangi bir katma değer de sağlamayacaktır. Buna ek olarak, elde edilen veriyi doğru uygulamalarla ile bir araya getirip, akıllı sistemler inşa etmek toplanan verideki saklı potansiyeli ortaya çıkaracaktır.

Dijitalleşmeyle birlikte elde edilen veriden fayda sağlanması gündeme geldiğinde, konu doğrudan makine öğrenmesi ve derin öğrenme alanına girmektedir. Bilgisayar dünyasındaki -özellikle son yıllarda tanık olduğumuz- gelişmelerin sonucunda bu alanlarda geliştirilen sistemler, daha uygulanabilir hale gelmiştir. Donanımsal alanda yeni işlemcilerin ve grafik işlemcilerin tanıtılması, yazılımsal alanda ise durum özelinde etkili ve verimli kütüphanelerin yayınlanması, bu kapsamda geliştirilen sistemlerin popülerliğinin artmasında önemli bir role sahip olmuştur.

Bu alanda yapılan çalışmalar dikkate alındığında, tavsiye sistemlerinin de bir alt alanı olan bir sonraki ürünün tahmini, zorlu ama bir o kadar da fayda sağlayan işlemler arasında yer almaktadır. Sahip olduğu müşterilerini iyi tanıyabilmek ve müşteriye özgü davranışları ve ilişkileri yüksek doğrulukla adresleyebilmek, bu alanda çalışmalarını ve işlerini yürüten kuruluşlar tarafından büyük bir önem atfedilen tavsiye

sistemleri için yüksek önceliğe sahip olmaktadır. Böylelikle, bu alanda öne çıkmayı başaran kuruluşlar aynı zamanda, bulundukları sektörde rakiplerine kıyasla bir adım öne geçmeyi başarabilmektedir. Bu amaç doğrultusunda, oturum tabanlı tavsiye sistemleri, araştırmacıların üzerinde çalışmalarını yoğunlaştırdığı ve tavsiye modellerini tanıttığı bir alan haline gelmiştir. Bu bakışa sahip bir tavsiye modeli göz önünde bulundurulduğunda, mevcut müşterinin bir oturum boyunca sergilediği aktivitelerden yola çıkılarak, aynı oturum süresince hangi ürün ile ilgilenebileceğinin tahminlemesi yapılabilmektedir.

Belirli bir tahminleme işlemini gerçekleştirmek üzerine çalışan tavsiye sistemleri, bu işlemi yüksek başarım oranıyla tamamlayabilmek için çok miktarda veriye ve bu veriyle birlikte özniteliğe ihtiyaç duymaktadırlar. Ancak, söz konusu tahminleme işlemi oturum tabanlı sistemler üzerinde gerçekleştirildiğinde, mevcut özniteliklerin azlığı sebebiyle önerilen çözüm yöntemlerini farklı bir bakış açısı gözeterek irdelemek gerekir. Bu tipteki veri kümelerinde herhangi bir profilleme işlemi olmadığından ve ürünler ile ilgili, yalnızca hangi üründen sonra hangi ürünün geldiği bilgisi olduğundan, geleneksel tavsiye modellerinde kullanılan yöntemler bu alanda yetersiz kalmaktadır. Bu sebeple, mevcut veriye ek özellik katarak, ürünler arasındaki ilişkileri daha doğru bir şekilde modelleyebilecek yöntemlerin bulunması, araştırmacılar arasında popüler bir konu haline gelmiştir. Bu alanda, son 25 yıldır önemli çalışmalar yapılmıştır. Son yıllarda yapılan çalışmalarda ise özellikle özyineli yapay sinir ağları tabanlı (RNN) çalışmalar dikkati çekmektedir. RNN yöntemlerinden birisi olan GRU4Rec modeli de öneri sistemleri üzerinde başarılı sonuç veren çalışmalardan birisidir. Ardından, NARM ve STAMP oturum tabanlı tavsiye modelleri ilgili alanda yürütülen çalışmaların sonuçları arasındadır. Son olarak, çizge sinir ağları alanında en gelişmiş kabul edilen tavsiye sistemi olan SR-GNN bu alandaki çalışmaların geldiği son nokta oalrak kabul edilmektedir. Bu çalışma kapsamında SR-GNN yönteminin iç yapısı incelenmiş olup, bazı alt birimleri üzerinde yapılan değişiklikler doğrultusunda testler yapılmış ve elde edilen sonuçların değerlendirilmesi yapılarak, yapılan değişikliklerin genel performansa etkisi ölçülmeye çalışılmıştır.

Gömme işlemi ile birlikte eldeki bir veri, daha toğun ve daha anlamlı bir şekilde ifade edilebilmektedir. Bu perspektif gözetilerek bir çok alanda başarılı sonuçlar elde edilmiştir. Benzer şekilde, yapılan çalışma dahilinde, mevcuttaki oturum bilgisi de gömme işlemi uyguladıktan sonra modele girdi olarak verilmektedir. Sonuç itibariyle, gömme birimi üzerinde yapılan değişikliğin orijinal modele olan etkisini daha doğru bir şekilde gözlemleyebilmek için, yürütülen deneylerde kullanılan veri kümeleri, orijinal modelin test edilmesi sırasında kullanılan veri kümeleriyle aynı olacak şekilde seçilmiştir. Bu bağlamda Yoochoose ve Diginetica olmak üzere, iki adet e-ticaret sayfasının sırasıyla 2015 yılında gerçekleştirilen ACM RecSys Challenge ve 2016 yılında gerçekleştirilen CIKM Cup yarışmalarda kullanılmak üzere sağlamış olduğu veri kümeleri üzerinde çalışılmıştır. Her iki veri kümesinde de anonim kullanıcılara ait hangi ürünler ile ilgilendiğini gösteren bir ürün listesi ve bu listedeki ürünler ile tam olarak hangi tarih ve saatte ilgilendiğini ifade edecek zaman damgası bilgisi bulunmaktadır. Bu bilgiler, eldeki verinin anlamlı bir şekilde oturum tabanlı tavsiye sistemine girdi olarak verilebilmesi için ilgili modelin ön işleme adımında kullanılmıştır.

Üzerinde deneylerin yapılan SR-GNN oturum tabanlı tavsiye modelinin, daha önce de bahsedildiği şekilde farklı işlerin yürütülmesi amacıyla tasarlanmış 3 farklı birimi bulunmaktadır. Bunlardan ilki olan ön işleme biriminde, testler sırasında kullanılan veri kümesinin, modele tarafından işlenmeye uygun hale getirilmesi işi

tamamlanmaktadır. Bu kapsamda, ham veri kümesinden elde edilen değerler, öncelikle her bir elemanın sahip olduğu zaman damgası bilgisine göre sıralı hale getirilmektedir. Bu sıralama işlemi sonrasında oturumda 5 üründen daha az ürün bulunan oturumlar veri kümesinden çıkarılmaktadır. Böylelikle model için ürünler ile ilgili yeteri kadar ilişki bilgisi taşımayan oturumlar elenmiş olup, işlemsel gücün gereksiz yere harcanmasının önüne geçilmesi amaçlanmıştır.

Oturumların gömme değerlerini hesaplamak için öncelikle elde edilen veri kümesindeki bilgiler kullanılarak, bu veri kümesindeki bütün ürünlerin yer aldığı bir bitişiklik matrisi elde edilmiştir. Her bir oturum birer alt çizge, bu oturumdaki her bir ürün bu çizgedeki bir düğüm ve ürünler arasındaki ilişki (birbiri ardına gelme vb.) birer kenar olarak ifade edilmiş ve bir çizge yapısı oluşturulmuştur. Bu yöntem gözetilerek kümülatif bir şekilde, tüm veri kümesinin çizge şeklinde ifade edilmesi sağlanmıştır. Bu noktada farklı oturumlarda gözlemlenebilecek aynı türden ilişkinin olduğu durumlarda mevcut ürünler arasına yeni bir kenar eklemek yerine, mevcuttaki kenarın ağırlığı birim kenar ağırlığı kadar artırılmıştır. Böylelikle iki ürünün arasındaki ilişkinin kuvvetinin, gömme işlemi sırasında da doğru bir şekilde ifade edilebilmesi amaçlanmıştır. Sonraki adımda, elde edilen gömme değerlerinin modelde bir sonraki adımda girdi olarak kullanılabilmesi için oturumda geçen ürünlerin gömme değerlerinin arasındaki mesafenin ifade edildiği bir matris oluşturulmuştur. Son olarak her iki gömme değeri arasındaki mesafeyi hesaplamak için, Öklid uzaklığından faydalanılmıştır. Bu noktada, orijinal yöntemde her iki gömme değerinin arasındaki mesafeyi hesaplarken Öklid uzaklığının kullanılması, yürütülen deneyler dahilinde de aynı uzaklığın kullanılmasında etkili olmuştur.

Gömme değerlerinin hesaplanması aşamasında, SR-GNN modelinin orijinal gerçeklemesinde, oturumların gömme değerlerini hesaplamak üzere temel bir gömme yöntemi kullanılmıştır. Bu işlem dahilinde gömme değerleri, her bir oturumun birer yönlü çizge ile ifade edildiği durum şeklinde düşünüldüğünde, bu çizgedeki kenarların ağırlıklarının normalize edilip, ortalamasının alınarak atanması şeklinde hesaplanmıştır. Tam olarak bu noktada, varsayılan gömme yöntemi yerine ürünler arasındaki ilişkileri daha iyi ifade edebileceği düşünülen 6 farklı gömme değeri kullanılarak modelin genel performansı artırılmaya çalışılmıştır. Oturumların gömme değerlerinin hesaplanması sırasında, oluşturulacak gömme vektörlerinin uzunluğu, yani her bir ürünün ifade edileceği öznitelik sayısı, ayarlanabilmektedir. Bu değişkenin optimum şekilde ayarlanmasının, oturumların gömme değerlerinin hesaplanması üzerinde büyük bir etkisi olduğunu ifade etmek gerekir. Yapılan deneyler dahilinde, elde edilen gömme değerlerinin uzunluğu 128 olacak şekilde ayarlanmıştır. Bu sayı seçilirken, word2vec gömme yöntemi her iki veri kümesi üzerinde 64, 128 ve 256 olmak üzere farklı uzunluklarda gömme vektörü üretecek şekilde çalıştırılmış, sonucunda elde edilen gömme değerleri tahminleme birimine girdi olarak verilmiştir. Her bir farklı uzunlukta gömme değeri için tahminleme modeli 3 kez çalıştırılmış ve elde edilen başarım değerlerinin ortalamaları karşılaştırılmıştır. Bu değerlerin arasından en yüksek başarıma sahip olan gömme değerleri 128 uzunluğa sahip vektörler ile ifade edildiğinden, kalan deneylerin yürütülmesi sırasında bu uzunlukta gömme değerleri kullanılmıştır.

Bu kapsamda tekli değer ayrıştırması (SVD), word2vec, node2vec, Deepwalk, laplacian eigenmaps ve LINE yöntemleri kullanılarak, iki farklı veri kümesi her biri 6 farklı şekilde incelenmiştir. Varsayılan yöntemde, ürünlerin gömme değerleri arasındaki Öklid mesafesi hesaplanmış olup, bu değerlerden oluşan matris tahminleme

işlemine gönderilmiştir. Çalışma dahilinde incelenen her bir gömme yöntemi için bu işlem o duruma özgü gömme değerleri kullanılarak tekrarlanmıştır.

Bu çalışmadaki asıl amaç, ürünlerin vektörler ile ifade edilirkenki kullanılan gömme yönteminin, orijinal model üzerindeki etkisini gözlemlemektir. Burada orijinal deney düzeneğinde kullanılan veri kümesinin sabit tutulmasıyla aynı sebepten yola çıkarak, incelenen yöntemlerin başarımını değerlendirmek için de orijinal yöntemde kullanılan başarım kriterleri kullanılmıştır. Bu bağlamda, incelenen modelin başarım değerleri Recall@20 ve MRR@20 performans kriterleri ile ölçülmüştür. Recall@20 kriteri, 20 elemanlık tavsiye kümesinde, gerçekte olması gereken ürünün bulunup bulunmadığına gör hesaplanırken, MRR@20 kriteri, gerçekte olması gereken ürünün, tavsiye kümesinde bulunduğu yerden uzaklığının hesaba katılmasıyla elde edilen değer ile ifade edilmektedir.

Deneylerden elde edilen başarım sonuçlarına göre yalnızca word2vec gömme yönteminin YOOCHOOSE veri kümesi üzerinde kullanıldığı durumda MRR@20 kriteri temel alındığında çok küçük bir başarım artışı gözlemlenmiştir. Bunun haricinde farklı gömme yöntemleri ve veri kümeleri kullanılarak elde edilen Recall@20 ve MRR@20 başarım değerleri, orijinal yöntemin başarım değerlerini geçmeyi başaramamıştır. Ancak bu değerlerin, orijinal modelin başarım değerlerinin çok az farklarla gerisinde kaldığı gözlemlenmiştir.

Sonuç olarak, oturumların gömme değerleri ile ifade edilmesi sırasında kullanılan yöntemin, kapılı çizge sinir ağı yapısı temelli tavsiye sistemi üzerinde etkili olduğu açık bir şekilde gözlemlenmiştir. Böylelikle, gömme değerlerinin hesaplanması sırasında kullanılacak yöntemin seçimi esnasında, üzerinde çalışılan veri kümesinin iç yapısının da incelenmesi ve çalıştırılacak gömme yöntemi seçilirken bunun da göz önünde bulundurulmasının gerekliliği ortaya çıkarılmıştır. Duruma özgü tavsiye modeli oluşturmak, gömme yönteminin seçilmesi konusundaki problemin giderilmesine çözüm olarak önerilmiştir.

# 1. INTRODUCTION

Machine learning and deep learning terms are getting involved in human's daily life by increasing rate day by day. Even if it is not directly noticed by people, the effects of these systems increased drastically in recent years. Most of the e-commerce and e-marketing websites, shopping applications, briefly, systems that collect a huge amount of data start to benefit from the power of data-oriented applications. Numerous fields are powered by machine learning and deep learning methodologies which are, image and audio processing, autonomous systems, natural language processing, entertainment, healthcare systems, marketing, and financial systems. Moreover, within each research area, there is a wide range of applications, and research studies are still carried out. Along with these studies, session-based recommender systems are one of the most popular and ever-growing fields that draw attention of the researcher community [1].

Within this study, a particular session-based recommendation model which is, Session-Based Recommendation System with Graph Neural Network, also called SR-GNN, was analyzed in detail by modifying some parts of the existing model. In the first step, a general overview of the recommender systems, session-based recommender systems, and embedding scheme which are the main titles that form the main context of the SR-GNN method are given. In compliance with this overview, numerous experiments were carried out to measure the effects of the modifications that have been applied to the overall SR-GNN model. In order to observe the effects of these modifications directly, the original experiment environment of the SR-GNN model is replicated as much as possible. To achieve this, the same datasets with the original execution environment which are YOOCHOOSE[1] and DIGINETICA[2] and have been used. According to the original recommendation model, SR-GNN, the main flow of this method consists of three modules; the first one is the pre-processing

[1] https://recsys.acm.org/recsys15/
[2] http://cikm2016.cs.iupui.edu/cikm-cup

module second one is the embedding module, the third and the last one is the neural network module. Within the context of this study, research questions have been focused on making modifications to the embedding module of the original SR-GNN model, specifically. Totally, 6 different embedding techniques have been applied to the embedding module of the model other than the default embedding technique of the original method. Each method has been executed with both introduced datasets. Sticking with the obtained results, a comprehensive analysis report has been presented as an outcome of this research. Lastly, proposals that can help to carry this research one step further have been discussed.

### 1.1 Purpose of Thesis

Recommender systems are considered as future of the e-commerce and e-marketing applications. Capability of recognizing their player base in a more compact and meaningful way will eventually result in being in the leading position among their competitors. In particular, improvements that will boost the performances of session-based recommender systems will unlock the way that leads these companies to be in such positions. In recent years, researchers are aimed to solve the problem of predicting succeeding items that comes after the particular sequence of items. Similar to these studies, the main purpose of the research is to investigate the correlation between the embedding technique applied within the SR-GNN method and prediction results. Focusing on this purpose, the following research questions were investigated within this study:

1- What are the most well-known and widely used recommender and session-based recommender models?

2- What is embedding in session-based recommender models?

3- How does the embedding module of the SR-GNN method affect the overall performance?

4- Is the selection of the embedding technique should be case-specific, depending on the applied dataset?

## 1.2 Related Works

Deep learning models along with the recommendation systems started to be a popular research area in the early 2000s. In particular, thanks to the developments and optimizations in the machine learning and deep learning libraries and the developments and hardware-based enhancements during the 2010s, researchers focused their studies on this specific field [2]. Within the effect of lean in the domain of the research area, various session-based recommendation methods have been introduced as a result of these studies.

One of the initial works on the recommendation systems is the POP/S-POP method simply offers to predict an item according to the number of re-occurrences of these items along with the specified session [3]. The technique basically adopts the perspective of statistical operations. This is the technique that is simply structured and executed in very short durations yet, it ignores the special relations between items all along the specific session. Eventually, it is a primitive method which has poor performance measures but brings a new glance to the field of recommendation systems by being an ancestor to the methods that will be introduced afterward in the field of session-based recommendation system.

Item-based K nearest neighborhood approach can also be applied as a solution to the session-based recommendation problems. According to the proposed model, unlike user-based collaborative filtering, this method performs the task of predicting the next item according to the similarity calculation between the item that will be predicted and the item set that the user has rated. According to the value of 'k', the next k number of items is recommended to the user [4]. As a calculation of similarity measurement, several approaches were adopted to perform the task of calculation of the similarity among items in Euclidean space. Inner product, cosine similarity, adjusted cosine similarity, and correlation-based (Pearson Correlation) similarity metrics are the most-used similarity metrics that are being used along with the item-based k nearest neighborhood recommender models. Basically, all mentioned similarity measurement methods, based on inner product operation in equation 1, where $A$ and $B$ are the vectors that have the length of $n$. Accordingly, cosine similarity is the normalized version of the inner product operation in equation 2 and, similarly, the correlation-based

similarity is the one that both normalized and centered version of the inner product operation in equation 3 where $X$ and $Y$ are the sets that have $n$ number of elements.

$$A \cdot B = \|A\|\|B\| \cos\theta \tag{1}$$

$$C\_sim = \frac{A \cdot B}{\|A\|\|B\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} {A_i}^2} \sqrt{\sum_{i=1}^{n} {B_i}^2}} \tag{2}$$

$$r_{XY} = \frac{\sum_{i=1}^{n} (X_i - \bar{X})(Y_i - \bar{Y})}{\sqrt{\sum_{i=1}^{n} (X_i - \bar{X})^2} \sqrt{\sum_{i=1}^{n} (Y_i - \bar{Y})^2}} \tag{3}$$

Bayesian personalized ranking model, also called BPR is another session-based recommendation approach that proposes there is no other model exists that focuses on optimization for the ranking of items, although, all these methods are specialized for the item prediction task. Therefore, the study of Bayesian personalized ranking introduces two new contributions which are generic optimization criterion, BPR-OPT, and learning algorithm learnBPR. BPR-OPT is a criterion obtained from maximum posterior prediction for maximizing user preferences. Bayesian formulation of correctly ranked items that will be used for user-specific recommendation is given in equation 4. $\Theta$, stands for vector representation of a class where $>_u$ is the hidden preference representation for specific user and $p(\Theta| >_u)$ notation represents the user-specific likelihood. learnBPR is an algorithm that aims to maximize the BPR-OPT criterion introduced within the same study. According to learnBPR, the standard gradient descent algorithm is inadequate, so, stochastic gradient descent algorithm is powered by a bootstrap sampling of training tuples [5].

$$p(\Theta| >_u) \propto p(\Theta| >_u) p(\Theta) \tag{4}$$

Factorizing Personalized Markov Chains (FPMC) is one of the earliest models that involve the sequence factor to the next item prediction. By making use of the basket history of the specific user, the item that the specific user is likely to buy tried to be predicted within this model. According to the introduced model, Markov chains have been executed on a personalized transition matrix which is generated from the users' basket history information. Sticking with its operation, it reveals the impact of the sequential information and long-term user preferences over the next item selection. To overcome the problem of the sparsity of transition probabilities that will be used for

the estimator, a new factorization model is introduced. In the end, reuniting the set of personalized Markov chains with the factorized transition cube introduces the factorized personalized Markov chain (FPMC) structure to be in the next item prediction task [6].

Depending on the recent developments and popularity shifts, almost every existing solution started to have reimplementation of itself by considering deep learning approach. GRU4REC is a methodology that uses recurrent neural networks (RNN) to active the task of predicting the next item along with the session-based data. According to the model, the input of the prediction network is the state of the specific session which is determined according to the result of equation 5. Benefitting from the gated recurrent unit model eliminates the vanishing gradient problem that comes with recurrent neural networks. In parallel with equation 5, $x_t$ term used as an input at the time $t$, $h_t$ represents the current state of the session, $z_t$ stands for the update gate of the gated recurrent unit in equation 6 while $\hat{h}_t$ given as a candidate activation function which is described in equation 7. Reset gate of the activation function which is nominated by $r_t$, is introduced within equation 8 [7].

$$h_t = (1 - z_t)h_{t-1} + z_t\hat{h}_t \tag{5}$$

$$z_t = \sigma(W_z x_t + U_z h_{t-1}) \tag{6}$$

$$\hat{h}_t = \tanh(W x_t + U(r_t \odot h_{t-1})) \tag{7}$$

$$r_t = \sigma(W_r x_t + U_r h_{t-1}) \tag{8}$$

Finally, the output of the prediction network becomes the prediction result for the next item along with the session. A general overview of the GRU4REC model is described in Figure 1.1 [7].

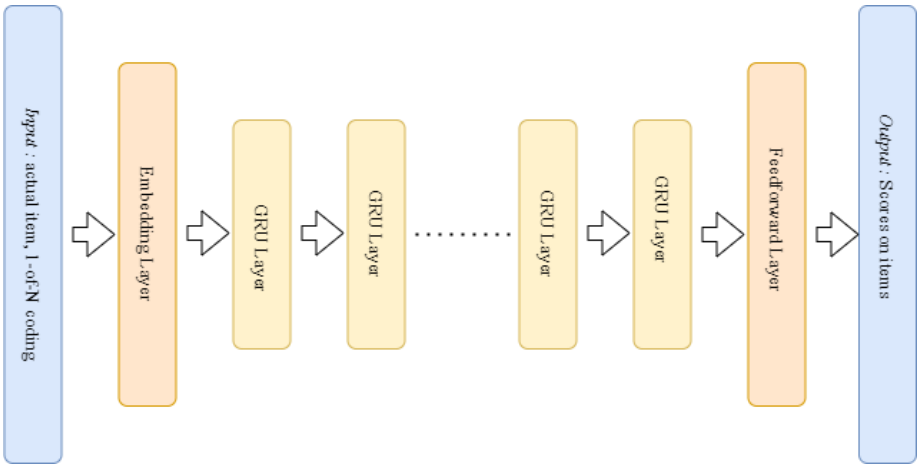


**Figure 1.1 :** GRU4REC main flowchart.

Neural Attentive Recommendation Machine (NARM), introduces an attention mechanism to the current RNN structure to be able to figure out the intention of the user along with the session. Similarly, with the GRU4REC, the model utilizes the recurrent neural networks for the task of the predicting next item. NARM is a session-based recommender model that brings encoder-decoder structures together and tries to find out the latent representation of the session. The main purpose of the model is to introduce a methodology that can cover both users' sequential behavior and the ultimate goal within the session. To achieve this, both global and local encoder mechanisms are used before the embedding layer of the current model. By using a global encoder, the sequential behavior of the user tried to be involved in the model while the main purpose of the user and most influential items that affect the users' behavior captured with local encoder structure. Session feature is a term introduced with this model and calculation of this term is described in equation 9 [8].

$$c_t = [c_t^g; c_t^l] = [h_t^g; \sum_{j=0}^{t} a_{ij} h_t^l] \tag{9}$$

Where $c_t^g$ and $c_t^l$ are the global and local attention values and $h_t^g$ stands for hidden state of the global encoder along with the specific session. $c_t^g$ value can be considered as $h_t$ notation in equation 5 and $c_t^l$ value can be obtained by using the following equation 10, where $a_{ij}$ in equation 11 emphasizes the factors that will be used or ignored and $h_j$ and $h_t$ are the hidden representations of the previous and current items, consecutively [8].

$$c_t^l = \sum_{j=1}^{i} a_{ij} h_j \tag{10}$$

$$a_{ij} = q(h_t, h_j) \tag{11}$$

$$q(h_t, h_j) = v^T \sigma(R_1 h_t + R_2 h_j) \tag{12}$$

Finally, obtained attention value is given as an input parameter to the recommendation model's ranking module and overall results are gathered [8].

The short-term attention/memory priority model (STAMP), can be considered as an extension model of the NARM which adds users' last click information to the model that is used in the NARM method. Along with the introduced model, the short-term memory priority model (STMP) is used to substantiate the last click event. According to the offered model, the STAMP method uses an attention network structure (Attention Net) which takes session embeddings and users' interest values as a vector, then, tries to model the behavior of the user to utilize the prediction task. Attention network built by two sub-modules, feed-forward neural network structure and attention function which are used for calculating attention weights for a specific item and calculating user-centric interest value that will be used for prediction, respectively. Calculation of attention coefficients vector given in equation 13 [9].

$$a_i = W_0 \sigma(W_1 x_i + W_2 x_t + W_3 m_s + b_a) \tag{13}$$

Where, $x_i$ is the ith item in a session, $x_t$ is the last click $W_0$ is weighting vector and $W_1$, $W_2$ and $W_3$ are weighting matrices. $b_a$ stands for bias factor and $m_s$ used for the representation of averaged external memory. Finally, attention-based user interest is calculated as it is stated in equation 14 by using attention coefficient values $a$, and the output value is used for the prediction task [9].

$$m_a = \sum_{i=1}^{t} a_i x_i \tag{14}$$

Session-based recommendation with graph neural networks, also called SR-GNN, fuses graph neural networks and session-based recommendation. SR-GNN model treats all sessions as directed graphs. Therefore, benefitting from the features of the graph structures is one of the key aspects of this method. According to the proposed

model, the main flow of the prediction task can be split into two sub-modules: embedding layer and neural network-based prediction layer. In the embedding layer, both items' and sessions' embedding values are calculated. For item embeddings, the model proposes to use gated graph neural networks [10]. Training of the vector representations of each node $v_i^t$ can be obtained by equation 15 and equation 16 [11].

$$v_i^t = \left(1 - z_{s,i}^t\right) \odot v_i^{t-1} + z_{s,i}^t \odot \tilde{v}_i^t \tag{15}$$

Where, $\tilde{v}_i^t$ represents the vector representation of the node i, $r_{s,i}^t$ in equation 17 and $z_s^t$ in equation 18 are the terms used for reset and update gates, consecutively. $\odot$ elementwise matrix multiplication operator $\sigma$ is the sigmoid function, $a_s^t$ is the matrix representation of a node that combines the incoming and outgoing edge representation matrices and $H$ is a weighting matrix in equation 19 [11].

$$\tilde{v}_i^t = \tanh\left(W_0 a_{s,i}^t + U_0\left(r_{s,i}^t \odot v_i^{t-1}\right)\right) \tag{16}$$

$$r_{s,i}^t = \sigma(W_r a_{s,i}^t + U_r v_i^{t-1}) \tag{17}$$

$$z_{s,i}^t = \sigma(W_z a_{s,i}^t + U_z v_i^{t-1}) \tag{18}$$

$$a_{s,i}^t = R_{s,i:}[v_1^{t-1}, \ldots, v_n^{t-1}]^T H + b \tag{19}$$

For session embeddings, $R_s$ the matrix is generated by combining the normalized incoming and outgoing edge weights of nodes in each session. According to the SR-GNN session embeddings used for representing long-term attention and combining these values with item embedding values will enable us to use them to make predictions about the next item inside the specific session [11].

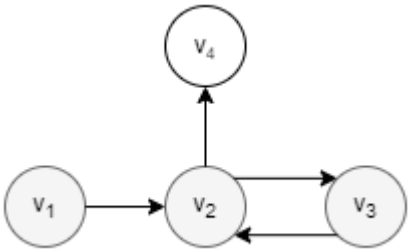


**Figure 1.2 :** Simple graph having 4 nodes and 4 edges in total [11].

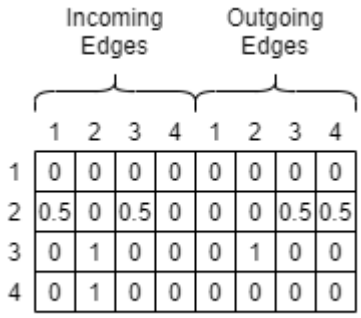


**Figure 1.3 :** Connection matrix of graph structure in Figure 1.2 [11].

Each session is treated as directed graphs. To process them within the prediction model each graph needs to be converted in a form that the neural network layer which is able to understand and execute an algebraic operation on that data. Graph structure in Figure 1.2 transformed into connection matrix of embedding values in Figure 1.3. To obtain the form that will be used in the next layer, Euclidean distances among each node are calculated with corresponding item embedding values and a final matrix filled with these calculated values for the specific session [11].

## 1.3 Proposed SR-GNN Comparison Architecture

According to the study within this research, the effect of the embedding type that has been used in the session embedding sub-module of the SR-GNN has been investigated. As it is implicated from the experimental result section, it is clear to say that the embedding type that has been used while learning the hidden features of sessions affects the performance of the session-based recommender model. Even if, examined models which use different embedding techniques are not able to overperform the vanilla embedding technique of SR-GNN, it is showed that it is possible to get better results by fine-tuning hyperparameters of the embedding algorithm. Moreover, experiments within this study have been executed on two specified datasets, YOOCHOOSE and DIGINETICA, and investigated embedding models may lead to expected results by boosting the vanilla recommender model.

A session-based recommender model that forms a baseline to the study in the context of investigating the effects of embedding types on the model consists of two major modules. The first module that named as embedding module is responsible for learning

hidden relationships among items and obtaining vector representations of these items considering these relationships. The second module is employed for making predictions according to these obtained vector representations of these items. According to the baseline model, in the embedding module, two types of embeddings have been introduced: item embeddings and session embeddings. After these embedding calculations are done by neural network-based structures, obtained values are fused and final embedding values are acquired. Lastly, the matrix that stands for representing Euclidean distance among these embedding values is calculated and given as an input to the succeeding layer. Within the scope of this study, we try to investigate the effect of the method that is picked for calculating session embedding values. According to the baseline model, each session is treated as directed graphs, items become nodes, sequential orderings of items are represented by edges among these items, and these embedding values are assigned as normalized edge weights among the nodes. As it is previously seen in Figure 1.2, the edge weights of each specific node are represented separately by considering incoming and outgoing edges. This approach has been accepted as session embeddings by the model and the prediction task has been performed accordingly. However, it is possible to throw out that there might be an alternative way that generates these session embeddings in a more meaningful and more compact way. At this point, this research investigates the effects of different embedding representations which are used to evaluate session embeddings on the overall recommendation task. During the experiments, six different embedding types which are singular value decomposition (SVD), large-scale information network embedding (LINE), word2vec, node2vec, Deepwalk, and Laplacian eigenmaps have been applied to the session embedding part of the baseline session-based recommendation model (SR-GNN). These experiment cases have been executed on two different datasets which are YOOCHOOSE and DIGINETICA. To measure the performance of examined models, the same evaluation metrics that are used on predecessor research studies have been used. Architecture diagram of proposed model is given in Figure 1.4.

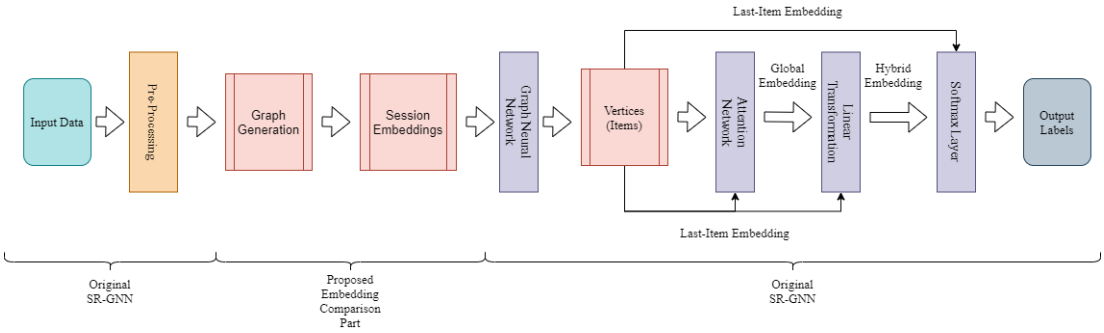


**Figure 1.4 :** Architecture diagram of the proposed model.

As a contribution of thesis, it is showed that changing embedding values of nodes belonging to the dataset, will affect the performance of the feed-forward network part of the SR-GNN model which is a state-of-art session-based recommendation model. According to the experiments that have been carried out within this research, it is possible to boost the performance of the model by choosing suitable, case-specific embedding techniques.

# 2. GENERAL OVERVIEW

## 2.1 Recommender Systems

Prediction of users' next behavior depending on their previous actions has been a very popular and also challenging task in the last recent years. The ability to process the data belongs to a specific individual in a way that guessing their next move becomes an essential process to achieve the prediction task. Within the developments of both hardware and software fields, this task has become easier and more efficient so that, researchers have become intrigued. On the hardware side, the sizes of the smallest components of both the central processing unit (CPU) and graphical processing unit (GPU) which is a transistor, became smaller and smaller and it led to major performance upgrades since, it becomes possible to squeeze more transistor inside the same area. On the software side, libraries that have focused on solving machine learning and deep learning problems efficiently and elegantly started to be used in a very large range and become very popular. In most cases, e-commerce, e-marketing websites benefit from these new developments to extend their customer base.

Luu et. al. define the relation between data, prediction, and recommendation properly in their research [12]. According to the definition, there exists a cycle between these three concepts that feed each other which is given in Figure 2.1. Data collected specifically to the case analyzed and some statistical operations performed (mean, variance, correlation calculations, etc.) on these data and machine learning, deep learning models have been trained based on this data. Then, in compliance with generated models, predictions have been made in order to provide the most acceptable one. These provided results become a recommendation for the specific case. For the last part considering the process cycle, recommendation results are fed to the data part for updating models according to their performance to create better models. With the last part offered process cycle has been closed and a self-feeding loop has been created.

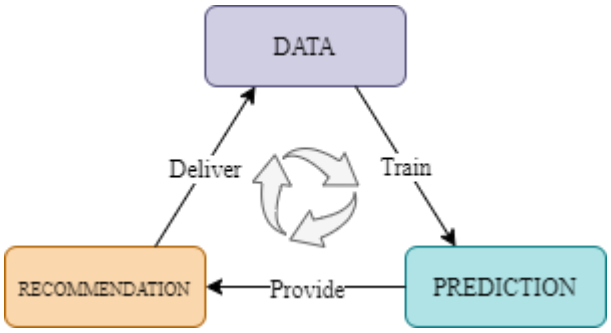


**Figure 2.1 :** Data, prediction, and recommendation cycle.

Recommender systems can be defined as a system that takes parameters from data that have been collected from related fields as an input and makes the most suitable predictions according to specific individuals as an output.

The recommender system can be investigated in detail by splitting it into three major categories. The first one is the content-based recommender systems which make recommendations by considering other users' preferences and tries to find out their resemblance with them. The second one is collaborative recommender systems that concern with making recommendations by considering the match between labels of items and users. The third and the last one is the hybrid recommender systems which benefit from these two introduced methodologies by creating a mixture of these models.

### 2.1.1 Collaborative recommender systems

There are few ways to make recommend an item to a specific user. One of the most common and widely used methodologies is the collaborative filtering technique. Within this methodology, items are proposed to a user by considering the similarities and differences with other users who are already inside the ecosystem. In Figure 2.2, a simplified diagram of the recommendation task with the collaborative filtering methods is given. The overall method relies on previous user-item interactions and the task of recommendation for new entries is completed upon these previous interactions. The quality of the recommendation highly depends on how well these previous interactions are learned. Depending on recommender model type, whether items are recommended to a user set or users are picked as compliant candidates for an item set. Since this method depends on relations between users and items it is crucial that having

a sufficient number of elements (users and items) for the performance of the generated recommendation model. The preference matrix that is used for this type of recommendation can be seen in Figure 2.3. In a similar way, collaborative recommender models work better with a large number of samples [13], so, at the beginning of the training phase, a cold start problem may arise. As it is stated in the description, a cold start problem was observed due to a lack of samples within the recommendation ecosystem. When there is not enough data exists to build a collaborative recommender system, it becomes inadequate to extract relations between users and items.

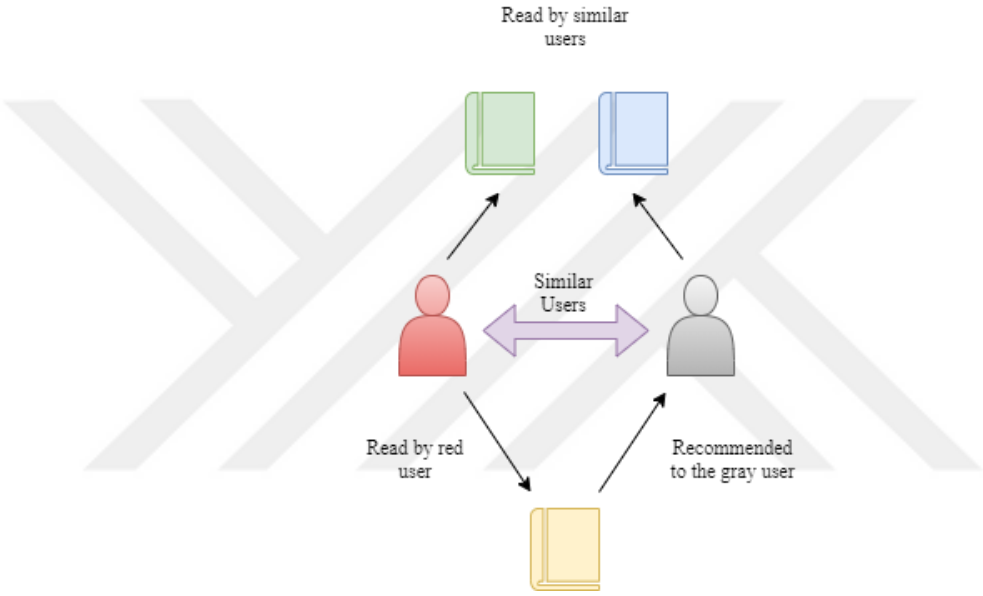


**Figure 2.2 :** Collaborative filtering.

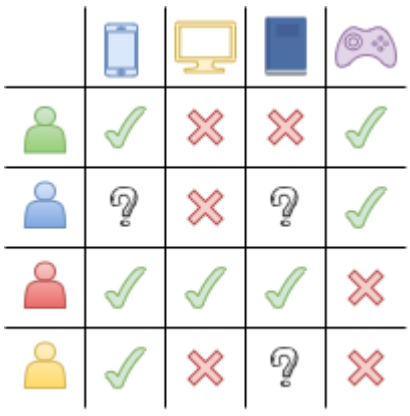

**Figure 2.3 :** Preference matrix of users.

Considering the collaborative filtering technique, there exist two approaches to apply which are memory-based collaborative filtering and model-based collaborative filtering.

#### 2.1.1.1 Memory-based collaborative filtering

Memory-based collaborative filtering highly depends on previous interactions and previous item ratings for making a new recommendation. According to this model, the first step is to find neighbors that have the same interests in common [14], [15]. After finding the neighbors, the remaining task is to pick one of the most suitable ones to make the prediction. There are several ways to determine which neighbor is the most similar and these are applied depending on the problem type. There exist two major approaches to apply memory-based collaborative filtering techniques. The user-based collaborative filtering approach tries to use similarity between users by considering their rating on the same item then, considering weighted average prediction ratings of the neighboring users to the target item, final prediction rating is calculated for active user on the target item. As the name implies that, the operation is user-centric and most of the calculations are completed from the user's perspective. Item-based collaborative filtering, on the other hand, tries to make calculations from the perspective of the items, not users. The main idea of this approach is to generate a model considering all ratings of an active user to predict the target item. This is achieved by how computing how similar are the users' previously rated items to the targeting item [16]. For both user-based collaborative filtering and item-based collaborative approaches, Pearson's correlation coefficient measurement and cosine similarity measurement techniques are mostly used.

Pearson's correlation formula given in equation 3, is used for calculating linear relation between two items, similarly, Pearson's correlation coefficient value represents the similarity between two items.

Cosine similarity is another technique used for calculating the similarity between two items. Other than Pearson's correlation calculation it is based on linear algebra instead of using some statistical calculations [16]. Within this method, items that their similarities will be compared, represented as vectors, and the similarity between these vectors are calculated by their lengths and angle between these two vectors. Basically, projection calculation of one vector to another is performed. Calculation of the cosine

similarity is given in equation 2. Where $A_i$ is the component of the vector $A$ and $B_i$ is the component of $B$ in equation 21.

#### 2.1.1.2 Model-based collaborative filtering

Unlike memory-based collaborative filtering recommender systems, model-based collaborative filtering recommender systems use pre-calculated models. According to this technique, a tailor-made model is built upon a specific problem by using a very similar dataset to the original one and it is expected to make successful predictions relying on generated model. During the model training phase, it is highly benefitted from machine learning and deep learning approaches. Since there will already be a pre-calculated model, the cold-start problem which occurs in memory-based collaborative filtering models will not show up in this case. Model-based collaborative filtering can be divided into three sub-groups which are clustering-based methods, matrix factorization-based methods, and deep learning-based methods. All these approaches can be used depending on the problem type in a flexible way.

In Table 2.1, memory-based collaborative filtering techniques and model-based techniques have been compared by considering their advantages and disadvantages.

**Table 2.1 :** Comparison of collaborative recommender models.

| Assessment | Memory-Based Collaborative Filtering | Content-Based Collaborative Filtering |
|---|---|---|
| Advantages | Easy to build<br>Scalable | Ability to operate on sparse data with dimensionality reduction |
| Disadvantages | Strongly affected by data sparsity<br>Not scalable | Latent factors block the inference |

### 2.1.2 Content-based recommender systems

Another well-known methodology to make a recommendation is the content-based recommendation. Content-based recommender systems make predictions according to the users' preferences as it is described in Figure 2.4. First of all, this kind of recommendation model tags items with certain keywords in order to address them. Then, new items tried to be predicted by using keywords of items that had already been preferred by active users. It is likely to observe that, there will be items having the same or similar attributes with previously selected items in the prediction set. Within

this model, prediction tasks are performed independently from other users' preferences. So, there will be no profiling process that exists in any steps of this methodology. Similarly, the cold start effect does not observe since each prediction is made by considering attributes of the items [16]. During the implementation of content-based recommender systems, there are numerous ways to find similarities between two items. Among these, mostly used techniques can be listed as, Term Frequency Inverse Document Frequency (TF/IDF), Naïve Bayes Classifier [17], Decision Trees [18], or Neural Networks [19]. All these kinds of similarity calculation techniques can be picked depending on the type of the problem.

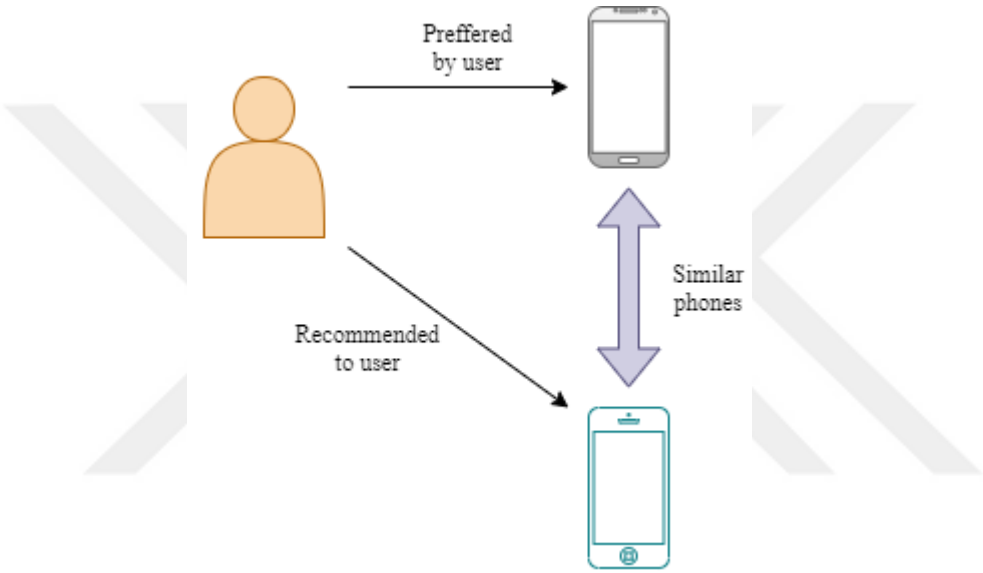


**Figure 2.4 :** Content-based recommendation.

### 2.1.3 Hybrid recommender systems

Hybrid recommender systems mostly bring together different types of recommender systems while benefitting from the advantageous parts of these methods. Surely, there exists some backing off points while combining these methods. Need to mention that, there will be a trade-off while using different methods from different conventions. The key point of building hybrid recommender methods that, compensating the disadvantages of one model by using another model. In this way, recommender models become available with higher accuracy and higher performance values [20]. It is possible to calibrate the mixture of different recommender models depending on the problem type. This situation allows the recommender system to be flexible since it is possible to accommodate itself. Hybrid recommender systems can be divided into 7

sub-group. Listings of approaches and descriptions of these approaches are given in Table 4 [21].

**Table 2.2 :** Hybrid recommender system types and definitions.

| Approach | Description |
|---|---|
| Weighting | To make a single recommendation, scores from different methods are brought together. |
| Switching | According to the case, the hybrid model switches between one of the self-forming methods. |
| Mixing | Recommendation scores from different types of recommender types are taken into consideration. |
| Feature Combination | Similar to the weighting approach, different techniques merged to make a recommendation. But in this case, features are used together to form a new approach rather than recommendation scores. |
| Cascading | The output of one recommender model is strengthened by another recommender model. |
| Feature Augmentation | Outputs of one recommender model, become an input value and processed by another model. |
| Meta-Level | The input of a recommender model becomes another recommender model upon training. |

## 2.2 Session-Based Recommender Systems

Session-based recommender systems are a specialized version of conventional recommender systems. Session term corresponds to a specific time interval that user interaction starts at a specific point and ends at a specific point. All interactions between system and user within a specific time interval -also called session- are now become data that can be processed in numerous ways. Yet, all these data gathered from a session is raw data that needs to be processed according to the problem type to be able to feed the recommender system.

From the perspective of session-based recommendation, the order or sequence of the items becomes an important feature. For most of the session-based recommender systems, after pre-processing step, the only data that remains is the sequential orderings of the items within a specific session considering their timestamp information. After that, the ability to represent this session in a different and more meaningful way becomes a key process that directly affects the performance of this type of recommendation model.

Unlike traditional recommender systems, generating profiles for a specific user is not mandatory in this case. It is possible to predict by analyzing the patterns among items that have been ordered sequentially. Considering the session-based recommendation systems, it is crucial to represent these items in a more meaningful way. After that, another key process with these systems is to reveal hidden relations among the items by applying special techniques. These are called embedding part and network part within the scope of session-based recommendation terminology and they have been observed in most of the session-based recommender models.

There exist several approaches to building a session-based recommender model. These can be split into three major categories which are traditional session-based recommender approaches, factorization-based approaches, and neural network-based approaches. For each category of session-based recommender approach, their sub-categories are listed under the relevant sections [22].

### 2.2.1 Traditional session-based recommender systems

It is possible to build a model considering the interactions between items performed by the user. During the training session, the most similar items inside the dataset were calculated and the prediction task was performed according to that prediction. This approach is one of the most widely used ones but it has limitations since the models solely depend on the last action within-session (prediction of the model is made upon the last item within that session). Three of the traditional session-based recommender models are pattern/rule mining based models, k-nearest neighbor based models, and Markov chain based models [22].

#### 2.2.1.1 Pattern/Rule mining based models

Pattern or rule mining-based models try to show relations among items by performing similarity calculations among them. These rules are mostly described by their togetherness within the specific session. These types of models are used with well-balanced and solid datasets. These models are beneficial for having an elementary structure and being effective in the cases that relations between items are relatively less complex. Yet, these methods are fragile with a lack of information inside the ecosystem [22].

#### 2.2.1.2 K-nearest neighbor based models

The prediction task of the newcomer item was performed by calculating the distances of the newcomer item to the nearest K number of items. Distance metric can be any kind of measure types such as Euclidean distance, Manhattan Distance, Minkowski distance, and Hamming distance. The class type of the newcomer item is recommended as the class label of the nearest item's class label. These models are not viable for complex datasets having sophisticated relations and orderings of the items does not affect the performance. It is also effective on such datasets with a rapid execution time. For generalization of the K-nearest neighbor-based methods, it is not always easy to decide value of K which represents the number of neighboring items. Also, it is not efficient to apply on complex datasets [22].

#### 2.2.1.3 Markov chain based models

Markov chain-based recommender systems are based on the Markov decision process (MDP), and these kinds of recommender models consider short-term effects on the specified session. Also, the possible output of the model is taken into consideration during the prediction process [23]. The datasets having long-term relationships inside long, unordered sessions are not in the practice area of Markov chain-based models. These models start to degrade as long as the session grows longer [22].

### 2.2.2 Factorization-based recommender systems

Factorization is another methodology that can be used for building a session-based recommendation system. It brings the ability of processing item features and session features while training the recommendation model. Moreover, this approach differs from the other ones by ceasing to use sequential order information among the items [24]. Types of recommender models that are based on factorization operation can be listed as latent factor-based models and distributed representation-based models [22].

#### 2.2.2.1 Latent factor-based models

Latent factor models mainly focus on discovering hidden features by using existing features. Moreover, data represented in high dimensional vector space is projected on low dimensional vector space. By this conversion, it is aimed to have a denser and more compact feature set. Latent Dirichlet Allocation (LDA) [25], restricted Boltzmann machines (RBM) [26], and singular value decomposition methods are well-

known methods within this field [27]. They can be applied for datasets having ordered sessions. It is a simple and effective method yet, mainly suffers from a lack of data and cannot perform meaningful inference from complex interactions [22].

#### 2.2.2.2 Distributed representation based models

Distributed representation-based models perform prediction tasks by assigning each item to a representational statement [28]. These statements can be numbers, vectors, or matrix depending on the problem type. This methodology aims to bring these representations together and try to merge them based on algebraic operations. For most cases, these models work better on unordered datasets and relation encoding abilities are powerful. Nevertheless, these systems fragile to noisy sessions and struggles to build prediction models by ordered data [22].

### 2.2.3 Deep neural network-based recommender systems

Deep neural network-based models try to solve the last item problem by involving the effect of the previous items onto the prediction. This problem is mostly observed in the previous categories of the recommender system approaches. The case-specific deep learning model needs to be trained and pre-calculated during the training phase. A kind of memory mechanism is used for building such recommender models and it is seen that performance of this type of systems boosted drastically [7]. There are numerous deep neural network-based recommender systems exists that can solve the problem of the prediction task within the session. These are; recurrent neural network (RNN) based models, multi-layer perceptron (MLP) based models, convolutional neural network (CNN) based models, graph neural network (GNN) based models, attention-based models, memory-based models, mixture models, generative models, and reinforcement learning-based models [22].

All these kinds of models build upon artificial neural network structure. These networks are composed of units called neurons having input values, output values, activation threshold, and weight values for each input signal. A simplified illustration of this unit is given in Figure 2.5. Activation of these units depends on the input weights and activation threshold value of this unit and according to these values, and an output value is generated and given as an input for another unit. The general structure of the artificial neural network consists of layers that grouping these neurons together. Considering the overall network, relying on the activated neurons, an output

value is generated. The activation function can be picked according to the problem type. Binary strep function (BSF), rectified linear activation (ReLU), sigmoid and tanh functions are used frequently as an activation function for these networks. The output generation task of the network is also called feed-forward. Depending on the correctness of the output value, the whole network got feedback, and the weights of the inputs are adjusted accordingly. This operation is also called back-propagation. Representation of a single artificial neuron structure is given in Figure 2.6.

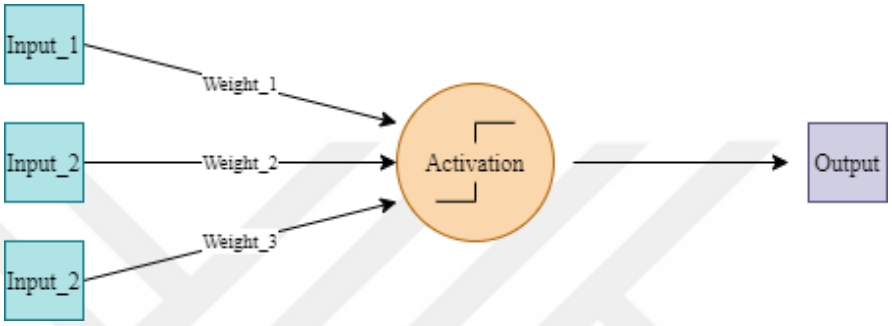


**Figure 2.5 :** Simplified representation of a single artificial neuron.

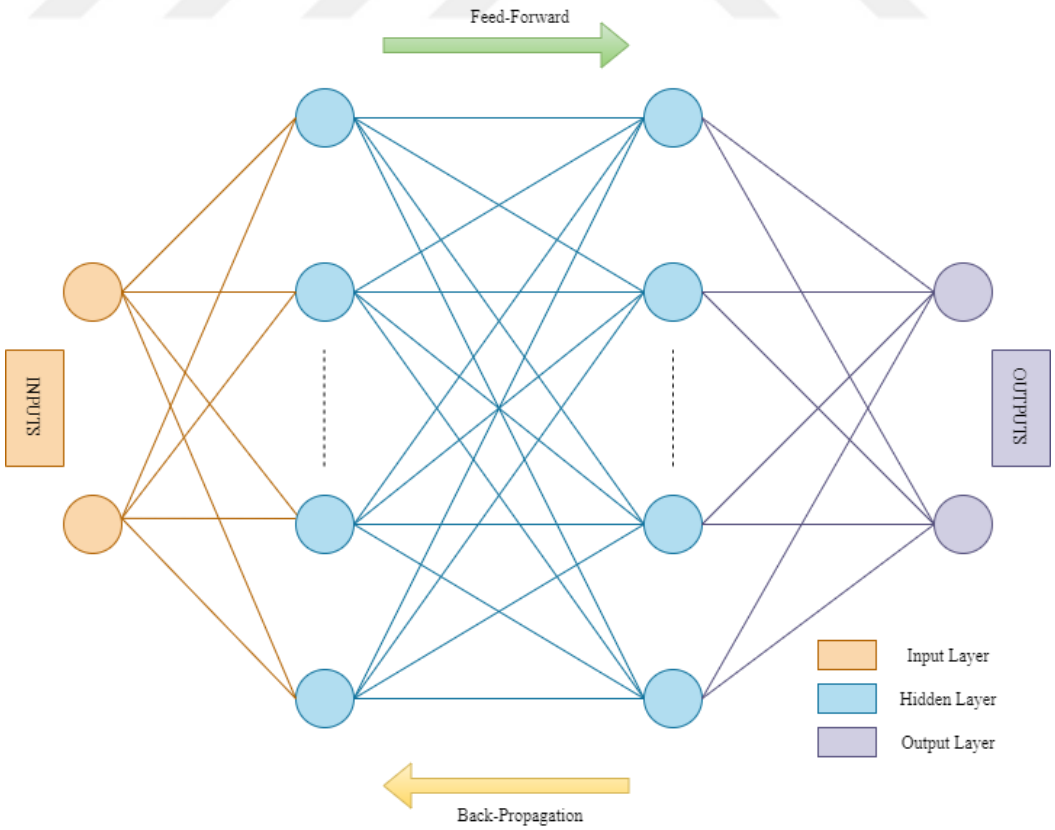


**Figure 2.6 :** General overview of an artificial neuron network structure.

#### 2.2.3.1 Multi-layer perceptron (MLP) based models

The multi-layer perceptron method is an application of artificial neural networks. During the prediction task, a different spectacle of session context learned and merged, eventually, the final prediction has been made according to this composition. These methods have three different layers which are the input layer, hidden layer, and output layer. For the special cases that having only one hidden layer inside their neural network structure, they are called vanilla neural networks [29]. Multi-layer perceptron models are not capable of processing the sequentially ordered sessions [22].

#### 2.2.3.2 Recurrent neural network (RNN) based models

Recurrent neural network systems consist of a group of artificial neural nodes that builds a network structure that uses sequential data and performs a prediction task accordingly. These networks build upon feed-forward networks and use a kind of memory structure to involve the past items' effects on successor ones [30]. Recurrent neural network-based models are mostly used on datasets where each session is relatively long and sequentially ordered. It is possible to learn complex relations among items inside the session by using recurrent neural network structures such as GRUs and LSTMs [22].

#### 2.2.3.3 Convolutional neural network (CNN) based models

Convolutional neural network-based recommender models are another application of the artificial neural network structure. These structures consist of specialized layers which are, convolutional layer, non-linearity layer, pooling layer, flattening layer, and fully connected layer. These layers provide the ability to learn features and classification operations to the overall network. Convolutional neural networks are mostly used on noisy sessions having sparse data within. These structures are durable for different cases but, they come with a higher time and space complexity [22]. Lastly, image and video processing applications mostly benefit from convolutional neural networks [31].

#### 2.2.3.4 Graph neural network (GNN) based models

Graph neural network-based recommender systems originated from representing each session as graphs and representing the relationships among items from this perspective. These models are also built upon artificial neural networks, moreover, they benefit

from the power of graph representation. Accordingly, each session turned into a graph structure and within this graph, each node represents an item from the session. Edges of these graph structures represent the relations between items. To process these graphs, all these nodes and relations need to be converted into a form that neural networks can perform algebraic operations and make predictions based on these values. Therefore, embedding operation which helps to transform these graphs into a processable format comes to aid. After an embedding operation -the type of embedding decided by problem case- performed on a graph, obtained representations, in other words, numbers, vectors, matrices, are given to the prediction module. Graph neural network-based models are divided into three sub-groups which are Gated graph neural networks (GGNN), convolutional graph neural networks (GCN), and graph attention networks (GAT). These models can operate on complex datasets having complex relationships among their items. As well as these models can represent latent features and relationships, their building and training cost relatively higher than average and the complexity of these models needs to be taken into consideration [22].

#### 2.2.3.5 Attention-based models

Attention-based models add an ability to represent the effect of the more relevant features to the final prediction. This task can be achieved in two ways, short-term attention and long-term attention. The concept of attention can also be considered as a memory mechanism. Determination of whether a memory is long term or short term is decided by two aspects: how long will the information is kept and how much data is stored inside the memory. Attention-based networks can be used on cases that having long-length sessions. While these networks can reveal the influential aspects inside the session, they cannot acquire sequential information from these sessions [22].

#### 2.2.3.6 Memory-based models

Memory-based methods propose an external memory structure to represent the relation between items. Memory structure which is used for prediction tasks provides an ability to carry the most relevant and influential information for the next item prediction. Memory-based models can operate on noisy and sparse datasets. While these models can keep the history of the previous information updated, they face challenging situations to reveal the sequential information [22].

#### 2.2.3.7 Generative models

Generative models focus on generation output values that cannot be discriminated from original input values. According to this approach, if it is possible to produce such values, it is also possible to predict the one that can come as the next item. When these models are investigated deeply, two different networks come up. While the first one -generator network- tries to generate the fake input values the second one -discriminator network- performs the task of determining whether an input value is generated or real. These models can be applied to cases that having dynamic and changing environments. These models have high time and space complexity, so, they can burden a huge load on the recommendation system [22].

#### 2.2.3.8 Mixture models

Mixture models bring a different variety of recommender models together to form a hybrid one. Following this paradigm, mixture models can capture different relations since each partition of the hybrid model specialized in capturing the different types of relations. As it is expected the more different types of models involved the higher performance value achieved. Similarly, as the number of different types of recommender models increases, the complexity of the hybrid model grows higher, respectively. Mixture models can work with heterogeneous datasets since they have a flexible structure. Thanks to the situation that comes from carrying advantageous aspects of different approaches, these models comply with different cases [22].

### 2.2.4 Evaluation of session-based recommendation model approaches

As is described previously, there exist three categories that divide session-based recommender models according to their attitudes. The selection of the recommender model is highly dependent on the specific case, so, there is no best approach to follow and any method can be useful relying on the case [32]. In general, the performance of the model is proportional to being flexible and adaptive to the specific situation. Nevertheless, there exist some facts that each kind of approach carry their own features. Traditional session-based recommender models tend to have a simpler, more straightforward structure than the other two categories. Moreover, they are easy to build and require less execution time to get results. They are expected to be having the least performance results compared to the other approaches, since, they are most effective on specific cases having datasets with less complicated relations with

themselves [33]. Another approach, which consists of factorization-based recommender models, is positioned in the middle of traditional session-based recommender models and neural network-based models considering overall performance measures. They have less computation cost than neural network-based models and for some specific cases they are able to overperform some kind of neural network-based methods such as recurrent neural network-based models [7]. The last approach which is formed by neural network-based methods is the one that has the highest performance results. This approach benefits from the neural network scheme which has become very popular in recent years. Within the developments in this field, performances of these models have grown upwards drastically [34]. It is not surprising that higher performance results come with a price. These methods are the ones that have the most complicated structure and require the utmost execution time among other types of models. Thanks to the complex structure, they have ability to learn hidden/latent features among items and by having this feature, they can overperform remaining models having different approaches [35].

### 2.2.5 Challenges & problematic cases with session-based recommender systems

There exists several challenging and also might be problematic topics when session-based recommender models are investigated deeply. First of all, the length of the session is one of the most important key attributes that have a direct impact on model performance. Considering datasets, long sessions are better compared to shorter ones, since, they are likely to carry more significant and more meaningful data. However, it gets harder to reveal the hidden relations as long as the session length grows bigger. Although longer sessions are mostly preferred over shorter sessions, in some cases, longer sessions carry more trivial relations which will not help to boost the performance of the recommender model [36]. On the other hand, shorter sessions are easy to process but carry fewer interactions. It is not always the case that, recommender systems can build a reliable model to predict the next item within shorter lengths. Data sparsity is another case that affects the performance of the prediction model. These kinds of models always need to have enough data to learn interactions between items adequately [37]. User information is also influential on the performance of the session-based recommender model. Considering this kind of recommendation system, for most cases, each session is treated as an individual session. So, sessions that are owned by the same client, processed as owned by different unique clients. This leads to missing

the opportunity of benefitting from this information. Hence, the model misses learning user-centric relations to obtain a more generalized session-based recommendation system. The last but not the least impactful concept on session-based recommender models is the sequential ordering of items. When the data is gathered from a specific domain, if data have any timestamp information, it is possible to order them based on their timestamp values. Thanks to this ordering operation, a relation among items become available to be learned and used by recommender models. If there is no timestamp information, it is not possible to extract any hidden relation based on orderings. Hence, it has no effect on the session-based recommendation model [36].

### 2.3 Embedding Scheme

Representing a real-world object in a way that computer systems can understand, means that digitalizing that object by identifying its features. This identification task can be achieved in various ways. One of the simplest ways to perform this, labeling its features by pre-defined keywords known by these systems. One hot encoding is a methodology used for representing these keywords within a feature matrix the way that machine learning and deep learning models can use. For instance, when, it is tried to represent a red box, it is also required to define its features such as its width, height, depth, color, etc. Each attribute of the item becomes a column inside the feature matrix belonging to the specific item. In a similar way with one-hot encoding, these digitalized terms also need to be represented as the way that machine learning and deep learning systems can understand and operate on them in an efficient way. Within this context, embedding operation comes to aid. By benefitting from this methodology, it is possible to represent an item in a more meaningful way. Also, during this operation, features of objects which are hard to detect at first sight also be revealed. Thanks to this information, these objects are represented with a piece of extra information that helps machine learning and deep learning systems while building models.

Embedding is a generalized term that corresponds to map the data from one vector space to another one. In most cases, this conversion is performed from higher dimensional vector space to lower-dimensional one. Additionally, it is possible to liken this operation to projecting the data to another vector space that has a different dimension. Key responsibility that needs to be taken into consideration is, there will not be any information loss observed, after, an embedding operation is performed on

data. Moreover, this operation might reveal new latent features that already exist within the data. In other words, the embedding task needs to preserve the internal structure of the dataset, while, adding new beneficial information.

### 2.3.1 Embedding in recommender systems

The task of recommendation is ranked among the fields that benefit from the embedding operation. In a similar way with other use cases of embedding, considering recommendation systems, each item needs to be represented in an efficient way that, these systems can perform operations that are required for the prediction task. Recommender systems that benefit from the embedding scheme and deep learning approach can be investigated within two sub-sections which are the embedding section and artificial neural network section. During the embedding section, the representation of each data piece is converted into another vector space, while, preserving its internal structure. Thanks to this operation, the same data is represented with fewer features which helps for boosting the overall performance. Moreover, depending on the embedding type that has been used for dimensionality reduction, new latent features may also be revealed which are hard to detect and process with a raw dataset. For the session-based recommendation systems, use cases of embedding operation mainly focus on converting graph-like structures into pre-defined sized vector representations. After that, each representation data are given as input values to the artificial neural network section. At the end of the neural network section, prediction values of the overall model are obtained.

As the name implies, datasets that have been used for session-based recommendation models consist of individual sessions. During embedding operation, each session is represented as graph structure considering their neighboring situations with other nodes. Considering this perspective, each graph represented with its adjacency matrix and embedding operation takes this adjacency matrix as an input value. Depending on the selection of the embedding technique and embedding size, each graph is represented with the desired type of embedding and size.

Embedding types can be investigated within three major categories which are graph adjacency matrix-based embedding methods, random walk-based embedding methods, and deep learning-based embedding models. Matrix factorization, Laplacian eigenmaps, Isomap, singular value decomposition, locally linear embedding (LLE),

high order proximity preserved embedding (HOPE), and GraRep methods are listed among matrix factorization based embedding methods, while, DeepWalk, node2vec, and struc2vec considered as random walk based embedding methods. In addition to this, word2vec, LINE, DNGR, GCN, SDNE, GraphGAN, Variational Graph Auto Encoders (VGAE), and GAE lie among the deep learning-based embedding methodologies.

### 2.3.2 Matrix factorization based embedding models

Graph adjacency matrix-based embedding models try to re-construct the target matrix into lower-dimensional vector space while keeping its internal structure unmodified or similar to the original representation. Also, this operation is aimed that having a more compact and dense structure than the original one. These models benefit from sophisticated matrix operations such as matrix decomposition, eigenvalue, and eigenvector calculation.

Matrix factorization is a sub-category of collaborative filtering-based methods that is widely used in recommendation systems. The main purpose of this method is to decompose the adjacency matrix which represents the relations among items into two lower-dimensional rectangular matrices [38].

Laplacian eigenmaps method projects data samples into lower-dimensional vector space by benefitting from eigenvectors of the input adjacency matrix. The main approach of this method is to preserve the local topology of the graph while representing these samples into another domain [39]. The objective function given in equation 20 tried to be minimized from the perspective of Laplacian eigenmaps. Where $X_i$ is the $i^{th}$ element $X_j$ is the $j^{th}$ element of set $X$, $W_{ij}$ is the weighting matrix and $\Phi(X)$ is the final embedding value of set $X$.

$$\Phi(X) = \sum_{i,j} \left| X_i - X_j \right|^2 W_{ij} \tag{20}$$

Isomap is one of the earliest methodologies among the graph adjacency matrix-based models, which refers to non-linear dimensionality reduction operation. Isomap algorithm tries to preserve the geodesic distance while reducing the dimension. According to this perspective, geodesic distance is defined as the sum of edge weights among node pairs considering the shortest path among them [40].

Singular value decomposition (SVD), is an operation that scales down the original vector space. Basically, singular value decomposition generates a matrix that each column corresponds to items, each row corresponds to users and each element within this matrix shows the relationship between these specific user and item pair. $M_{mxd}$ in equation 21, is the matrix that will be deconstructed by performing SVD method. Accordingly, three different matrices are generated as it is seen in the equation 22 and these matrices correspond to the user's hidden features which are $U$ in equation 21, the effect of each feature showed by $\Sigma$ in equation 21 and item's hidden features represented as $V^T$ in equation 21. Finally, the obtained matrix which is $mxd$, reduces the overall dimensionality and shows the relationships between users and items [41]. Compact form of the matrice $M_{mxd}$ is also given in equation 23.

$$M_{mxd} = U\,\Sigma\,V^T \tag{21}$$

$$M_{mxd} = \begin{bmatrix} u_{11} & \cdots & u_{m1} \\ \vdots & \ddots & \vdots \\ u_{1k} & \cdots & u_{mk} \end{bmatrix} \begin{bmatrix} \sigma_{11} & & \\ & \ddots & \\ & & \sigma_{kk} \end{bmatrix} \begin{bmatrix} v_{11}^T & & v_{k1}^T \\ & \ddots & \\ v_{1d}^T & & v_{kd}^T \end{bmatrix} \tag{22}$$

$$M_{mxd} = \sigma_1 u_1 v_1^T + \sigma_2 u_2 v_2^T + \cdots + \sigma_k u_k v_k^T \tag{23}$$

Locally linear embedding (LLE), is an operation that allows reducing the dimensionality of a non-linear dataset while safeguarding its original geometric attributes. According to this method, firstly, K-nearest neighboring samples are located. Then, each sample represented as a linear combination of its neighbors by taking their weights into account. For the last step, it calculates the weights that could reconstruct the data sample by its neighboring vectors and generates lower-dimensional projections using these weights [42].

High order proximity preserved embedding (HOPE), describes a methodology that is used for revealing asymmetric transitivity in large-scale graph-like structures while saving the original high-order distances among data points [43].

GraRep is an embedding model that learns the node representations on weighted graph-like structures. Accordingly, it is aimed to obtain lower-dimensional vector representations from the original graph structure. Moreover, GraRep tries to involve the effect of the global structure during this projection process [44].

### 2.3.3 Random walk based embedding models

DeepWalk is a dimensionality reduction approach that learns hidden features among the nodes within the graphs. According to the methodology, firstly, any node has been randomly picked from a graph. Then, randomized traversals have been executed over the existing graph with pre-defined path lengths. Finally, the process of repeating this procedure to all existing nodes will generate the overall embedding values of each node within the graph [45].

The node2vec, aims to shrink down the dimensionality of the nodes belonging to a graph into lower spaces while trying to preserve the neighborhood structure of the original graph by maximizing the objective function given in equation 24 where, $u$ is the source node, $N_s(u)$ represents the network neighborhood of node $u$ and $f$ is the mapping function from nodes to feature representations. Mainly, this approach benefits from breadth-first search and depth-first search algorithm to gather node sequences [46].

$$\sum_i logPr(N_s(u)|f(u)) \tag{24}$$

During the process of finding similarities between each node pairs inside the graph structure, struc2vec approach focuses to find these similarities by using the network topology of the overall graph considering neighboring nodes. According to the model, the first step is to determine the structural similarity among nodes. Using these similarities, a multi-layered graph has been generated by considering their structural similarities. Then, a random-walking process is executed on this multi-layered graph structure. Finally, hidden relations inside the generated node sequences which are fed by a multi-layered graph are obtained. Thanks to these obtained hidden relationships, it becomes possible to represent each node by using fewer features [47].

### 2.3.4 Deep learning based embedding models

The word2vec, as the name of the method implies, tries to represent the words by vectors. This methodology forms a baseline for other types of random-walk-based methods by its approach to the problem. Word2vec can be performed by executing two different methods which are the common bag of words (CBOW) and skip-gram. They are both supported by the neural network approach and for the ultimate goal,

they both try to represent these words in a lower-dimensional space by using more meaningful and sparse features [48].

Large-scale Information Network Embedding (LINE) is an artificial neural network-based method that is used for data visualization, edge prediction and classification of nodes from the perspective of mapping the features in a lower-dimensional space. The method itself can efficiently be scaled with the increasing number of nodes inside the graph and can operate on directed, undirected, and weighted graphs. During the dimensionality reduction operation, the model tries to keep the local structure and global structure. To overcome the performance and efficiency problems of predecessor methods, an edge sampling algorithm was proposed with this approach [49].

Unlike, DeepWalk, which forms a baseline method to Deep Neural Network for learning graph Representation, also called DNGR, this approach uses a random walk model powered by artificial neural network structure instead of sampling the nodes within the graph in order to create node sequences. Additionally, the main purpose of the method remains the same, obtaining smaller-sized vectors that represent the nodes within the graph structures [50].

Graph convolutional network, GCN, is the generalized name for artificial neural network structures that benefits from convolution operation while transforming the target features in a more compact and dense space. Basically, the GCN term stands for the convolution operation that is applied on graph-like structures, instead of matrices. Benefiting from this method, it is expected from this neural network structure to learn latent features of a graph by involving the neighboring nodes and preserve the structure of the overall network.

Structural Deep Network Embedding (SDNE), proposes a semi-supervised neural network-based model composed of different non-linear network organizations. This allows the proposed approach to cover the hidden relations within the non-linear network structure. To preserving the structure of the overall network, first-order proximity and second-order proximity terms are introduced within this method. Accordingly, first-order proximity is used by a supervised module of the proposed method while second-order proximity is used by an unsupervised module to preserve the general structure of the network during the operation of dimensionality reduction [51].

GraphGAN is a powerful dimensionality reduction operation that builds upon the same argument with graph adversarial networks. Similar to GANs, GraphGAN also has two modules that, one is trying to generate the fake samples and the other one tries to find out whether the produced input is fake or not. In a similar way, GraphGAN operates on graph structures regarding this perspective. The generative module of the model gets a specific node and generates a graph adhering to its idiosyncratic features such as its neighboring nodes and edge weights. Generated graph structure is given to the discriminative module of the model and its reality is questioned. During this operation, it is possible to reveal new hidden features, data points can be represented within fewer features [52].

Autoencoders are kind of unsupervised artificial neural network-based structures that consist of two sub-operation which are encoding and decoding. In a similar manner, these methods have three layers which are input layer, an output layer, and a hidden layer. The hidden layer of this network is the essential part that latent features of an input value are learned. Following the overall process in steps, firstly, the input value is transferred to the hidden layer and the output value is generated relying on edge weights of the hidden layer. These edge weights are adjusted according to the result of the output layer. Also, the size of the hidden layer is hyperparameter which is adjustable to the problem case. Graph autoencoders (GAE) and variational graph autoencoders (VGAE), is a specialized version of autoencoder implementation that takes graphs as input values and operates by considering the edge weights and neighboring nodes of these graphs. Both approaches try to represent each input graph more compactly by benefitting from unseen features learned by autoencoder networks [53].

# 3. EXPERIMENTAL RESULTS

## 3.1 Experiment Overview

To observe the effect of the embedding method that has been used in the GNN based session-based recommendation model, several experiments have been carried out. These experiments can be divided into three major steps which are pre-processing, embedding, and prediction steps.

In the pre-processing step, YOOCHOOSE and DIGINETICA datasets have been converted into a more compact form. Both original dataset files include the click history information that has been gathered from corresponding web pages. Accordingly, in each dataset file, there exist 4 different fields for each data point which are session-id value, timestamp information, item id value, and item category value. During the pre-processing phase, these click events are grouped by their session-id values, and considering the specific session, each click event is ordered by its timestamp values. Lastly, timestamp values have been removed and only item id values in an ordered manner remain for a specific session. After pre-processing step, generated data file becomes ready to use by the recommender model part.

After the pre-processing step is completed, generated values are used by embedding the module of the GNN based session-based recommender model. As is described before, the baseline model, SR-GNN, has two embedding sub-modules which are item embeddings and session embeddings. For item embedding sub-module, random values are assigned to each item and these randomized embedding values are trained through a neural network-based structure. For the session-embedding sub-module, each session is treated as sub-graphs, and a complete graph of the whole dataset is generated by adding these sub-graphs together. Then, the embedding process has been executed on the adjacency graph of the complete graph structure via its adjacency matrix. Within the vanilla GNN based model, embedding values of sessions are calculated by benefitting from adjacency matrices, where relationships (or edges) between items are represented as normalized edge weights within these adjacency matrices. According

to the research question of this study, there might be a better, more meaningful, and compact way to represent these sessions. To investigate the answer to this question, the default embedding technique of the vanilla model has been replaced with different embedding techniques. After this sub-step, session embedding values merged with item embedding values as it is handled in the vanilla session-based recommender model.

For the last step, the Euclidean distance of each node pairs is calculated by forming a matrix structure and this matrix is used for the prediction task while training the overall recommender model. Similarly with item embedding sub-step, artificial neural network structure has been used to adjust the edge weights among the hidden layers of this neural network structure, during the prediction task. The overall flow diagram can be examined in Figure 1.4.

### 3.2 Experiment Environment (Hardware & Software)

Experiments that have been carried out by this study require a specific execution environment and specific software installations. According to these requirements following settings have been used during the phase of the experiment:

*Hardware Specifications:*

**CPU** **:** Intel (R) Core (TM) i7-8700k CPU @ 3.70 GHz (12 CPUs), ~3.7GHz

**GPU** **:** NVIDIA GeForce GTX 1070

**VRAM** **:** 8088 MB

**RAM** **:** 16384 MB

**Operating System** **:** Windows 10 Education 64 Bit

*Software Requirements:*

Python 3.7.2

Tensorflow 1.9.0

PyTorch 0.4.0

### 3.3 Experiments

In compliance with the purpose of this study, a sub-project has been carried out to obtain the specified embedding values of the sessions stated in the YOOCHOOSE and DIGINETICA datasets. To achieve this task Python implementations of singular value decomposition (SVD), large-scale information network embedding (LINE), word2vec, node2vec, Deepwalk, and Laplacian eigenmaps methodologies have been integrated through the vanilla SR-GNN implementation.

During the pre-processing step, trivial sessions that having items less than 5 have been removed. Before the calculation of the embedding vectors of each item, complete the graph structure of the dataset generated for both YOOCHOOSE and DIGINETICA datasets. For each case, the adjacency matrix of the complete graph has been generated. During this step, if item A is a successor item of item B in a specific session, there will be an edge from node A to node B in the graph. Similarly, with the adjacency matrix representation of the complete graph its assigned value of 1 to the corresponding cell. If there exists another relationship inside the different session, instead of adding a new edge to the complete graph of the dataset, its weight increased by one. Again, in the adjacency matrix, the corresponding cell value increased by one for each new edge appears from node A to node B. This perspective allows embedding operation to involve the effects of items that reside together but take place in different sessions. Following this methodology, the graph adjacency matrix of a complete dataset has been generated and became ready for extracting vector representations of items.

While calculating these embedding values, the size of the embedding vectors is adjusted by fine-tuning the corresponding embedding size hyper-parameter. Heuristically, 3 different embedding size parameters which are 64, 128, and 256, have been tried with the whole recommendation model and continued with the parameter value which gives the best results among these. Word2vec embedding technique has been executed 3 times on each YOOCHOOSE and DIGINETICA dataset to generate the session embedding values having the length of 64, 128, and 256 vector representations. The prediction results of these executions can be investigated in Table 3.1. Accordingly, the best result has been observed with the embedding size of 128, since then, the remaining experiments held out with an embedding size of 128 for each specific run.

**Table 3.1 :** Performance results of the word2vec implementation during the session embedding.

| word2vec embedding size through SR-GNN | YOOCHOOSE RECALL@20 | YOOCHOOSE MRR@20 | DIGINETICA RECALL@20 | DIGINETICA MRR@20 |
|---|---|---|---|---|
| 64 | 66.41 | 27.32 | 44.67 | 12.24 |
| 128 | 69.65 | 30.44 | 47.88 | 15.21 |
| 256 | 68.21 | 29.11 | 46.39 | 13.11 |

After the embedding values have been calculated, these values are used to create a matrix that represents Euclidean distance among each embedding pair, since, the original method takes this parameter as an input for the prediction layer. Lastly, this distance matrix passed through a neural network structure that is responsible for the item prediction. The whole recommendation model has been trained by repetitively executing this network structure.

### 3.4 Datasets

During the investigation study about the effects of the embedding type over the session-based recommender model, two different datasets have been used which are YOOCHOOSE and DIGINETICA datasets. The motivation for the selection of these datasets comes from two major causes. First, each individual dataset has been reached its maturity by being used on one of the top-rated challenges in this field, which is ACM RecSys Challenge (YOOCHOOSE) in 2015 and CIKM Cup (DIGINETICA) in 2016. Moreover, predecessor session-based models have been used the YOOCHOOSE and DIGINETICA datasets in their experiments. To make a better comparison between these models and investigated cases, the same datasets have been picked intentionally.

As it is described earlier, each dataset has been pre-processed before executing recommender models. To make the obtained datasets more reliable and consistent, items that appear less than 5 times in a session have been removed. This step allows the model to work on sessions that are likely to contain meaningful but hidden relationships among items. After this step has been applied, 7,981,580 sessions and 37,483 items were observed in the YOOCHOOSE dataset, while, 204,771 sessions and 43097 items were included by the DIGINETICA dataset.

During the experiment session, data within the YOOCHOOSE dataset became too much to handle by exceeding the memory limit of the experiment environment. To avoid this problem, the most recent 1/64 part of the YOOCHOOSE dataset has been used for succeeding tests as the same strategy within predecessor studies.

Along with this study, all 'YOOCHOOSE' dataset keyword corresponds to 1/64$^{th}$ portion of the original YOOCHOOSE dataset.

### 3.5 Evaluation Metrics

Along with this study, unlike traditional machine learning and deep learning models performance criteria such as accuracy, precision, recall and F1 score more specialized versions of these evaluation criteria which are MRR@K and Recall@K criteria have been used.

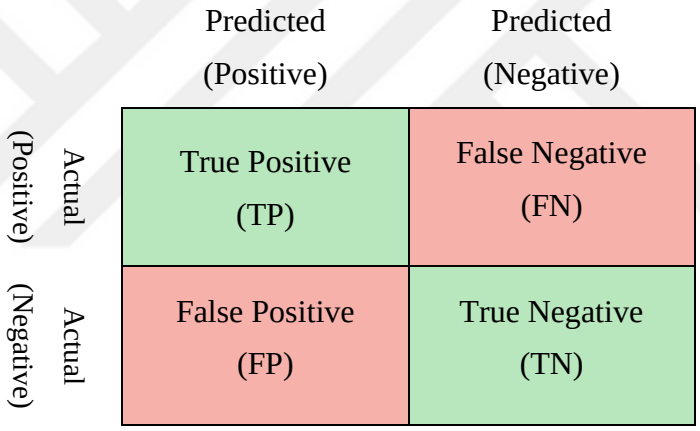


**Figure 3.1 :** Confusion matrix.

*True Positive* : Predicted and prediction is true

*True Negative* : Not predicted and prediction is true

*False Positive (Type 1 Error)* : Predicted but the prediction is wrong

*False Negative (Type 2 Error)* : Not predicted but the prediction is wrong

According to the confusion matrix (Figure 3.1), calculations of accuracy, precision, recall, and F1 score metrics can be represented by the equation 25, equation 26, equation 27, equation 28 respectively.

$$accuracy = \frac{TP + TN}{TP + FP + TN + FN} \tag{25}$$

$$precision = \frac{TP}{TP + FP} \tag{26}$$

$$recall = \frac{TP}{TP + FN} \tag{27}$$

$$f1\ Score = 2 * \frac{precision * recall}{precision + recall} \tag{28}$$

As it is described earlier, a more complex version of these evaluation metrics which are MRR@K and Recall@K, have been used within this research. Two key points lie behind the reason why these metrics have been used. Firstly, these metrics represent more meaningful results from the perspective of recommendation systems. Then, predecessor models that form the baseline to this study have used these MRR@K and Recall@K metrics. To make a better comparison of the cases investigated with this study and predecessor ones, the same evaluation metrics have been used.

Recall@K in equation 29 represents the ratio of correctly predicted items inside the top K items.

$$Recall@K = \frac{n_{hit}}{N} \tag{29}$$

Where, $n_{hit}$ represents the number of correctly predicted items that have been observed inside the the predictiction set which has *K* elements.

Mean Reciprocal Rank @K (MRR@K) in equation 30, an average of the reciprocal ranks of the items that correctly predicted. Values above K are set as 0.

$$MRR@K = \frac{\sum_{i=1}^{N} Rank(i) = \begin{cases} \frac{1}{x}, x \leq K \\ 0, x > K \end{cases}}{N} \tag{30}$$

K is an arbitrary value that can be chosen according to the case. Within this study, this value was picked as 20 since the quality of the model is compared with its predecessors.

### 3.6 Experimental Results

Within this research, to evaluate the effect of the session embedding technique over the whole recommender model tried to be investigated by conducting several experiments. Totally 12 different cases were investigated by using 6 different

embedding types during performing the session embedding within two different datasets. Each of these executions was performed three times and the average of these executions are assigned as a result of this specific case. According to these results, it is seen that changing the embedding type during the calculation of the session embedding affects the overall performance of the vanilla GNN based recommender model.

Within these set of executions, it is observed that only 1 execution which is the word2vec embedding method has been performed during session embeddings has exceeded the MRR@20 performance metric of the original SR-GNN model on the YOOCHOOSE dataset (Figure 3.3). The remaining executions are not able to overperform the vanilla model, but, all these collected results remain very close to the performance results of the original SR-GNN by ranging between 68.66% - 69.89% Recall@20 rate in Figure 3.2 and 29.69% - 30.68% MRR@20 rate in Figure 3.3. In terms of Recall@20 criteria, all test cases with different embeddings overperformed by the original SR-GNN with a 70.27% Recall@20 rate. Among the other embeddings, word2vec has followed the vanilla model by having a 69.89% Recall@20 rate. Performance results of the remaining embedding methods are represented in Table 3.2.

**Table 3.2 :** Performance results of the different embedding implementations during the session embedding sub-module on the YOOCHOOSE dataset.

| Embedding Implementation | RECALL@20 | MRR@20 |
|---|---|---|
| Original SR-GNN | 70.27 | 30.65 |
| SVD | 69.27 | 30.14 |
| LINE | 69.05 | 29.80 |
| word2vec | 69.89 | 30.68 |
| node2vec | 69.06 | 29.69 |
| DeepWalk | 69.11 | 29.80 |
| Laplacian EigenMaps | 68.66 | 30.16 |

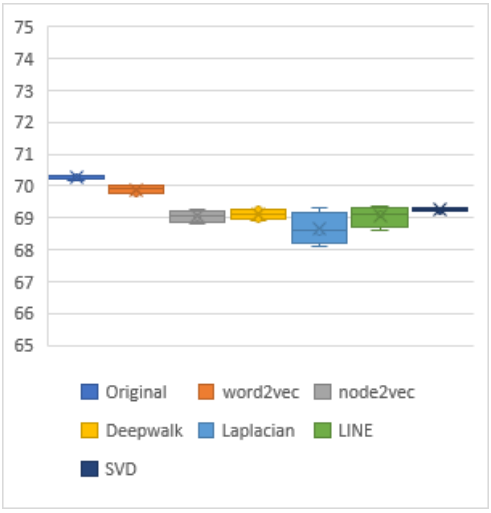


**Figure 3.2 :** Recall@20 values of YOOCHOOSE dataset.

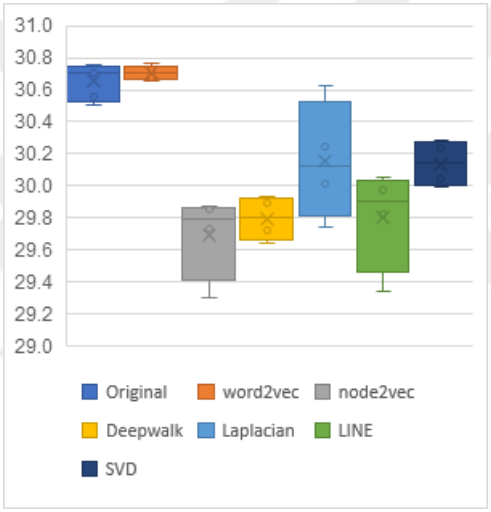


**Figure 3.3 :** MRR@20 values of YOOCHOOSE dataset.

Similarly, for the test cases that performed on the DIGINETICA dataset, no specific implementations of embedding types were achieved to overperform the vanilla SR-GNN method. Laplacian eigenmaps embedding method has performed very similar results with the vanilla model by having a 48.45% Recall@20 and 16.13% MRR@20 rate, which is just below the vanilla SR-GNN by 1.14% and 0.32% respectively. The remaining test results range between 48.09% - 47.05% in terms of Recall@20 rate in Figure 3.4 and 15.40% - 14.96% MRR@20 rate in Figure 3.5. which are still close to the vanilla SR-GNNs performance measures. Overall performance results of each test case are listed in detail in Table 3.3.

**Table 3.3 :** Performance results of the different embedding implementations during the session embedding sub-module on the DIGINETICA dataset.

| Embedding Implementation | RECALL@20 | MRR@20 |
|---|---|---|
| Original SR-GNN | 49.59 | 16.45 |
| SVD | 47.37 | 15.28 |
| LINE | 47.05 | 14.96 |
| word2vec | 48.09 | 15.40 |
| node2vec | 47.63 | 15.10 |
| DeepWalk | 47.52 | 15.23 |
| Laplacian EigenMaps | 48.45 | 16.13 |

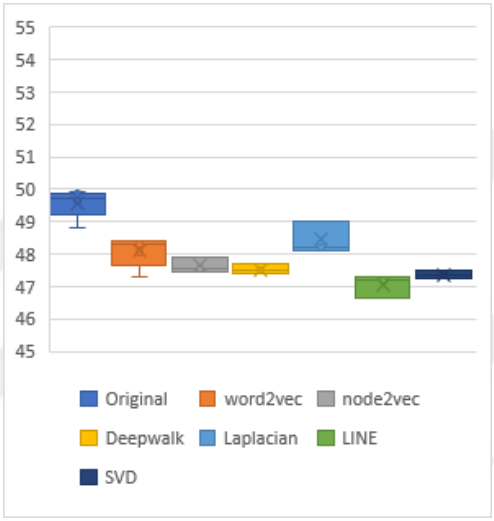


**Figure 3.4 :** Recall@20 values of the DIGINETICA dataset.


**Figure 3.5 :** MRR@20 values of DIGINETICA dataset.

## 4. CONCLUSION AND FUTURE WORK

Along with this study, the effect of the embedding type that is used during the session embedding sub-module of the GNN based recommender model has been investigated. To achieve this task totally six different embedding types which are singular value decomposition, Laplacian eigenmaps, Deepwalk, node2vec, word2vec, and LINE have been applied datasets to obtain vector representations of each session inside the overall dataset. Following the flow of the investigated model, these embedding values generated by using the session information are combined with the item embedding values as a default operation of the vanilla model. Then, thanks to these merged embedding values, the prediction network generates a recommendation vector that includes number of 'K' items. Finally, Recall@K and MRR@K performance values are calculated by the state that, actual item occurs in the recommended item set which has number of 'K' elements inside (Within the scope of this study K value is assigned as 20). All these experiments have been carried out on two different datasets which are YOOCHOOSE and DIGINETICA.

According to the results gathered during these tests, for the YOOCHOOSE dataset, it is observed that only the model that having word2vec session embedding, performed better than the vanilla SR-GNN method in terms of MRR@20 criteria. In the remaining test scenarios, there is no other model exists that having better performance criteria than the original SR-GNN method considering both Recall@20 and MRR@20 performance metrics. For the DIGINETICA dataset, the original SR-GNN model overperformed all of the models that have different session embeddings. Moreover, Laplacian eigenmaps become the embedding type that can sidle up to the vanilla SR-GNN model in terms of Recall@20 and MRR@20 performance metrics.

Considering the execution principles of the word2vec method, it tries to perform a prediction task based on learning the hidden relationships among items by considering the sequential ordering information of the items. Therefore, session length becomes proportional to the prediction performance meaning that the longer the session becomes, the better recommendation results obtained. When the pre-processed

datasets are analyzed in detail, it is observed that the YOOCHOOSE dataset has a longer average session length than the DIGINETICA dataset. Respectively, the YOOCHOOSE dataset has an average of 7.02 items in each session while DIGINETICA has an average of 5.22 items. Hence, the reason why the word2vec embedding method showed better performance on the YOOCHOOSE dataset -even having head-to-head performance with original SR-GNN- is likely to be derived from the average session length attribute. It might be possible that using the model with word2vec session embedding on a dataset having a longer average session length attribute may result in observing far better performance values within the recommender model having the word2vec embedding method.

Laplacian eigenmaps embedding method tries to preserve the local structure of the graph while performing the embedding task. Considering the geometrical structure of the YOOCHOOSE dataset, edges among the items rally on specific nodes, while, edges in the DIGINETICA dataset spread in a wider range of nodes. Considering a graph, if two edges have the same or similar edge weights, then, embedding values of the nodes that are connected by these edges also become the same or similar, as a result of the characteristics of the Laplacian eigenmaps embedding technique [54]. Therefore, Laplacian eigenmaps likely to shows better performance on preserving the local structure of the session while projecting the session into different vector spaces. In a similar perspective for the word2vec case, the highlight the effectiveness of the Laplacian eigenmaps embedding method over the datasets having more balanced structure, the new dataset that having the desired characteristics can be tested.

As it is described in earlier sections, during the embedding calculation operation, the embedding size was assigned as 128 by conducting several experiments. Accordingly, by using the word2vec method having 64, 128, and 256 length vector representations, the best results were obtained by the model that having 128 length vectors. As a future work, fine-tuning this parameter with smaller steps has been added to research plan to investigate its effect on overall performance results.

## 5. REFERENCES


[1] **Sarker, I. H. (2021)**. Machine Learning: Algorithms, Real-World Applications and Research Directions. SN Computer Science, 2(3), 160. https://doi.org/10.1007/s42979-021-00592-x

[2] **Thompson, N. C., Greenewald, K., Lee, K., & Manso, G. F.** (2020). The Computational Limits of Deep Learning. arXiv:2007.05558 [cs, stat]. Retrieved from http://arxiv.org/abs/2007.05558

[3] **Konow, R., Tan, W., Loyola, L., Pereira, J., & Baloian, N.** (2011). A Visited Item Frequency Based Recommender System: Experimental Evaluation and Scenario Description. J. UCS, 17, 2009-2028.

[4] **Altman, N. S.** (1992). An Introduction to Kernel and Nearest-Neighbor Nonparametric Regression. The American Statistician, 46(3), 175–185. https://doi.org/10.1080/00031305.1992.10475879

[5] **Rendle, S., Freudenthaler, C., Gantner, Z., & Schmidt-Thieme, L.** (2009). BPR: Bayesian Personalized Ranking from Implicit Feedback, 10.

[6] **Rendle, S., Freudenthaler, C., & Schmidt-Thieme, L.** (2010). Factorizing personalized Markov chains for next-basket recommendation. Içinde Proceedings of the 19th international conference on World wide web - WWW '10 (s. 811). Program name: the 19th international conference, Raleigh, North Carolina, USA: ACM Press. https://doi.org/10.1145/1772690.1772773

[7] **Hidasi, B., Karatzoglou, A., Baltrunas, L., & Tikk, D.** (2016). Session-based Recommendations with Recurrent Neural Networks. arXiv:1511.06939 [cs]. Retrieved from http://arxiv.org/abs/1511.06939

[8] **Li, J., Ren, P., Chen, Z., Ren, Z., & Ma, J.** (2017). Neural Attentive Session-based Recommendation. arXiv:1711.04725 [cs]. Retrieved from http://arxiv.org/abs/1711.04725

[9] **Liu, Q., Zeng, Y., Mokhosi, R., & Zhang, H.** (2018). STAMP: Short-Term Attention/Memory Priority Model for Session-based Recommendation. İçinde Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (ss. 1831-1839). New York, NY, USA: Association for Computing Machinery. https://doi.org/10.1145/3219819.3219950

[10] **Li, Y., Tarlow, D., Brockschmidt, M., & Zemel, R.** (2017). Gated Graph Sequence Neural Networks. arXiv:1511.05493 [cs, stat]. Retrieved from http://arxiv.org/abs/1511.05493

[11] **Wu, S., Tang, Y., Zhu, Y., Wang, L., Xie, X., & Tan, T.** (2019). Session-based Recommendation with Graph Neural Networks. Proceedings of the

AAAI Conference on Artificial Intelligence, 33, 346-353. https://doi.org/10.1609/aaai.v33i01.3301346

[12] **Luu, V.-T., Forestier, G., Ripken, M., Fondement, F., & Muller, P.-A.** (2016). Web usage prediction and recommendation using web session clustering (ss. 107-113). https://doi.org/10.1109/ICDIM.2016.7829779

[13] **Zhao, X.** (2016, January 28). Cold-Start Collaborative Filtering (Doctoral). Doctoral thesis, UCL (University College London). Doctoral program name: University College London, UCL (University College London). Retrieved from https://discovery.ucl.ac.uk/id/eprint/1474118/

[14] **Zhao, Z.-D., & Shang, M.** (2010). User-Based Collaborative-Filtering Recommendation Algorithms on Hadoop. İçinde 2010 Third International Conference on Knowledge Discovery and Data Mining (ss. 478-481). Program name: 2010 Third International Conference on Knowledge Discovery and Data Mining. https://doi.org/10.1109/WKDD.2010.54

[15] **Zhu, X., Ye, H., & Gong, S.** (2009). A personalized recommendation system combining case-based reasoning and user-based collaborative filtering. İçinde 2009 Chinese Control and Decision Conference (ss. 4026-4028). Program name: 2009 Chinese Control and Decision Conference. https://doi.org/10.1109/CCDC.2009.5192712

[16] **Isinkaye, F. O., Folajimi, Y. O., & Ojokoh, B. A.** (2015). Recommendation systems: Principles, methods and evaluation. Egyptian Informatics Journal, 16(3), 261-273. https://doi.org/10.1016/j.eij.2015.06.005

[17] **Friedman, N., Geiger, D., & Goldszmidt, M.** (n.d.). Bayesian Network Classifiers, 33.

[18] **Duda, R. O., Hart, P. E., & Stork, D. G**. (n.d.). Part 1: Pattern Classification, 69

[19] **Bishop, C. M.** (n.d.). Pattern Recognition and Machine Learning, 9.

[20] **Schafer, J. B., Frankowski, D., Herlocker, J., & Sen, S.** (2007). Collaborative Filtering Recommender Systems.

[21] **Burke, R.** (2002). Hybrid Recommender Systems: Survey and Experiments. User Modeling and User-Adapted Interaction, 12. https://doi.org/10.1023/A:1021240730564

[22] **Wang, S., Cao, L., Wang, Y., Sheng, Q. Z., Orgun, M., & Lian, D.** (2021). A Survey on Session-based Recommender Systems. arXiv:1902.04864 [cs]. Retrieved from http://arxiv.org/abs/1902.04864

[23] **Shani, G., Brafman, R. I., & Heckerman, D.** (n.d.). An MDP-Based Recommender System, 8.

[24] **Hidasi, B., & Tikk, D.** (2016). General factorization framework for context-aware recommendations. Data Mining and Knowledge Discovery, 30(2), 342-371. https://doi.org/10.1007/s10618-015-0417-y

[25] **Blei, D. M., Ng, A. Y., & Jordan, M. I.** (2003). Latent dirichlet allocation. The Journal of Machine Learning Research, 3(null), 993-1022.

[26] **Smolensky, P.** (1986). Information processing in dynamical systems: foundations of harmony theory. İçinde Parallel distributed processing: explorations in the microstructure of cognition, vol. 1: foundations (ss. 194-281). Cambridge, MA, USA: MIT Press.

[27] **Van Loan, C. F.** (1976). Generalizing the Singular Value Decomposition. SIAM Journal on Numerical Analysis, 13(1), 76-83. https://doi.org/10.1137/0713009

[28] **Le, Q. V., & Mikolov, T.** (2014). Distributed Representations of Sentences and Documents. arXiv:1405.4053 [cs]. Retrieved from http://arxiv.org/abs/1405.4053

[29] **Hastie, T., Tibshirani, R., & Friedman, J.** (2009). The Elements of Statistical Learning: Data Mining, Inference, and Prediction, Second Edition (2. bs). New York: Springer-Verlag. https://doi.org/10.1007/978-0-387-84858-7

[30] Annual Reviews in Control. (n.d.). Retrieved from https://www.journals.elsevier.com/annual-reviews-in-control

[31] **Browne, M., & Ghidary, S.** (2003). Convolutional Neural Networks for Image Processing: An Application in Robot Vision (ss. 641-652). https://doi.org/10.1007/978-3-540-24581-0_55

[32] **Parimi, R., Trepka, T., Caragea, D., & Bennett, C.** (2015). How to Choose a Recommender System: Insights and Experiences for Large-Scale User Personalization. https://doi.org/10.1109/BigDataCongress.2015.76

[33] **Ludewig, M., Mauro, N., Latifi, S., & Jannach, D.** (2019). Performance comparison of neural and non-neural approaches to session-based recommendation. İçinde Proceedings of the 13th ACM Conference on Recommender Systems (ss. 462-466). New York, NY, USA: Association for Computing Machinery. https://doi.org/10.1145/3298689.3347041

[34] **Kayid, A., Khaled, Y., & Elmahdy, M.** (2018). Performance of CPUs/GPUs for Deep Learning workloads. https://doi.org/10.13140/RG.2.2.22603.54563

[35] **Zhang, S., Yao, L., Sun, A., & Tay, Y.** (2019). Deep Learning based Recommender System: A Survey and New Perspectives. ACM Computing Surveys, 52(1), 1-38. https://doi.org/10.1145/3285029

[36] **Hu, L., Cao, L., Wang, S., Xu, G., Cao, J., & Gu, Z.** (2017). Diversifying Personalized Recommendation with User-session Context. Içinde Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence (ss. 1858-1864). Program name: Twenty-Sixth International Joint Conference on Artificial Intelligence, Melbourne, Australia: International Joint Conferences on Artificial Intelligence Organization. https://doi.org/10.24963/ijcai.2017/258

[37] **Meng, W., Yang, D., & Xiao, Y.** (2020). Incorporating User Micro-behaviors and Item Knowledge into Multi-task Learning for Session-based Recommendation. Proceedings of the 43rd International ACM SIGIR

Conference on Research and Development in Information Retrieval, 1091-1100. https://doi.org/10.1145/3397271.3401098

[38] **Koren, Y., Bell, R., & Volinsky, C.** (2009). Matrix Factorization Techniques for Recommender Systems. Computer, 42(8), 30-37. Program name: Computer. https://doi.org/10.1109/MC.2009.263

[39] **Belkin, M., & Niyogi, P.** (2001). Laplacian eigenmaps and spectral techniques for embedding and clustering. İçinde Proceedings of the 14th International Conference on Neural Information Processing Systems: Natural and Synthetic (ss. 585-591). Cambridge, MA, USA: MIT Press.

[40] **Tenenbaum, J. B., Silva, V. de, & Langford, J. C.** (2000). A Global Geometric Framework for Nonlinear Dimensionality Reduction. Science, 290(5500), 2319-2323. https://doi.org/10.1126/science.290.5500.2319

[41] **Klema, V., & Laub, A.** (1980). The singular value decomposition: Its computation and some applications. IEEE Transactions on Automatic Control, 25(2), 164-176. Program name: IEEE Transactions on Automatic Control. https://doi.org/10.1109/TAC.1980.1102314

[42] **Roweis, S. T., & Saul, L. K.** (2000). Nonlinear Dimensionality Reduction by Locally Linear Embedding. Science, 290(5500), 2323-2326. https://doi.org/10.1126/science.290.5500.2323

[43] **Zhu, D., Cui, P., Zhang, Z., Pei, J., & Zhu, W.** (2018). High-Order Proximity Preserved Embedding for Dynamic Networks. IEEE Transactions on Knowledge and Data Engineering, 30(11), 2134-2144. Program name: IEEE Transactions on Knowledge and Data Engineering. https://doi.org/10.1109/TKDE.2018.2822283

[44] **Cao, S., Lu, W., & Xu, Q.** (2015). GraRep: Learning Graph Representations with Global Structural Information. İçinde Proceedings of the 24th ACM International Conference on Information and Knowledge Management (ss. 891-900). New York, NY, USA: Association for Computing Machinery. https://doi.org/10.1145/2806416.2806512

[45] **Perozzi, B., Al-Rfou, R., & Skiena, S.** (2014). DeepWalk: online learning of social representations. İçinde Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining (ss. 701-710). New York, NY, USA: Association for Computing Machinery. https://doi.org/10.1145/2623330.2623732

[46] **Grover, A., & Leskovec, J.** (2016). node2vec: Scalable Feature Learning for Networks. İçinde Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (ss. 855-864). New York, NY, USA: Association for Computing Machinery. https://doi.org/10.1145/2939672.2939754

[47] **Ribeiro, L. F. R., Savarese, P. H. P., & Figueiredo, D. R.** (2017). struc2vec: Learning Node Representations from Structural Identity. Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 385-394. https://doi.org/10.1145/3097983.3098061

[48] **Mikolov, T., Chen, K., Corrado, G., & Dean, J.** (2013). Efficient Estimation of Word Representations in Vector Space. arXiv:1301.3781 [cs]. Retrieved from http://arxiv.org/abs/1301.3781

[49] **Tang, J., Qu, M., Wang, M., Zhang, M., Yan, J., & Mei, Q.** (2015). LINE: Large-scale Information Network Embedding. İçinde Proceedings of the 24th International Conference on World Wide Web (ss. 1067-1077). Republic and Canton of Geneva, CHE: International World Wide Web Conferences Steering Committee. https://doi.org/10.1145/2736277.2741093

[50] **Cao, S.** (2016). deep neural network for learning graph representations.

[51] **Wang, D., Cui, P., & Zhu, W.** (2016). Structural Deep Network Embedding. Içinde Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (ss. 1225-1234). Programme name: KDD '16: The 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco California USA: ACM. https://doi.org/10.1145/2939672.2939753

[52] **Wang, H., Wang, J., Wang, J., Zhao, M., Zhang, W., Zhang, F., … Guo, M.** (2017). GraphGAN: Graph Representation Learning with Generative Adversarial Nets. arXiv:1711.08267 [cs, stat]. Retrieved from http://arxiv.org/abs/1711.08267

[53] **Kipf, T. N., & Welling, M.** (2016). Variational Graph Auto-Encoders. arXiv:1611.07308 [cs, stat]. Retrieved from http://arxiv.org/abs/1611.07308

[54] **Goyal, P., & Ferrara, E.** (2018). Graph Embedding Techniques, Applications, and Performance: A Survey. *Knowledge-Based Systems*, *151*, 78–94. https://doi.org/10.1016/j.knosys.2018.03.022

**CURRICULUM VITAE**

**Name Surname** **:** Mustafa Aker

**EDUCATION** **:**

- **B.Sc.** **:** Department of Computer Engineering, Istanbul Technical University, 2016
- **M.Sc.** **:** Department of Computer Engineering, Istanbul Technical University, 2021

**PROFESSIONAL EXPERIENCE AND REWARDS:**

- 05.2017 – 01.2018, Middleware Engineer at EnerjiSA
- 01.2018 – 05.2021, Researcher & Software Engineer at TUBITAK BILGEM
- 05.2018 – Present, Software Engineer at Siemens

**PUBLICATIONS, PRESENTATIONS, AND PATENTS ON THE THESIS:**

- **Aker, M., Yildiz, C., & Yaslan, Y.** (2021). A Comparison of Different Embedding Methods on SessionBased Recommendation with Graph Neural Networks. 11th International Conference on Pattern Recognition Systems (ICPRS2021). Curicó, Chile, March 17-19.

**OTHER PUBLICATIONS, PRESENTATIONS AND PATENTS:**

- **Yildiz, C., Aker, M., & Yaslan, Y.** (2021). Next Item Prediction Using Neural Networks with Embedding Initialized Weights. IEEE EUROCON 2021-18th International Conference on Smart Technologies. Lviv, Ukraine, July 6-8.