ÇUKUROVA UNIVERSITY
INSTITUTE OF NATURAL AND APPLIED SCIENCES

PhD THESIS

Hakan YILMAZER

DEVELOPMENT OF RECOMMENDER SYSTEM ALGORITHMS FOR COLD-START PROBLEM

DEPARTMENT OF COMPUTER ENGINEERING

ADANA, 2022

ÇUKUROVA UNIVERSITY
INSTITUTE OF NATURAL AND APPLIED SCIENCES

DEVELOPMENT OF RECOMMENDER SYSTEM ALGORITHMS FOR COLD-START PROBLEM

HAKAN YILMAZER

PhD THESIS

DEPARTMENT OF COMPUTER ENGINEERING

We certify that the thesis titled above was reviewed and approved for the award of the degree of the Philosophy of Doctorate by the board of jury on 19/07/2022.

…….....	……...	…….…….....
Prof. Dr. Selma Ayşe ÖZEL	Prof. Dr. Umut ORHAN	Asst.Prof. Dr. Alper Kamil DEMİR
SUPERVISOR	MEMBER	MEMBER
……	…….	
Assoc.Prof. Dr. Mehmet Uğraş CUMA	Assoc.Prof. Dr. İrem ERSÖZ KAYA	
MEMBER	MEMBER	

PhD Thesis is written at the Department of Computer Engineering of Institute of Natural and Applied Sciences of Çukurova University
Registration Number:

Prof. Dr. Sadık DİNÇER
Director
Institute of Natural and Applied Sciences

ABSTRACT

PhD THESIS

DEVELOPMENT OF RECOMMENDER SYSTEM ALGORITHMS FOR COLD-START PROBLEM

Hakan YILMAZER

ÇUKUROVA UNIVERSITY
INSTITUTE OF NATURAL AND APPLIED SCIENCES
DEPARTMENT OF COMPUTER ENGINEERING

Supervisor : Prof. Dr. Selma Ayşe ÖZEL
Year: 2022, Pages: 131
Jury: : Prof. Dr. Selma Ayşe ÖZEL
: Prof. Dr. Umut ORHAN
: Asst. Prof. Dr. Alper Kamil DEMİR
: Assoc. Prof. Dr. Mehmet Uğraş CUMA
: Assoc. Prof. Dr. İrem ERSÖZ KAYA

Cold-start problems are one of the most important challenges in recommendation systems. In this thesis, we proposed models to develop solutions for the cold-start problem from two different perspectives. We aimed for a deterministic and a heuristic study that can be used in different scenarios. In the first perspective, we introduced a new heuristic framework that optimizes item-based similarity models to provide top-N recommendation lists using Continuous Ant Colony Optimization with a non-deterministic approach. Thanks to its heuristic structure, we aimed to create specific recommendation lists for users and change them according to each session, while at the same time aiming to balance the relevance of the user and the item variety in the recommendation lists. In the second perspective, we introduced two new Collaborative Filtering techniques deterministically. In the first model, we developed an asymmetric similarity matrix among the items based on the z-score normalization of the Gram-matrix we obtained using the implicit data, and in the second model, we aimed to reduce the sparsity with the item predictions with the assist our novel item similarity matrix, thus enabling more accurate decomposition of the latent factors in the user-item matrix we provided. We evaluated all of our methods on well-known datasets and observed that our methods outperform similar recommendation models in a variety of scenarios, including cold-start users, cold-start systems, and providing of unpopular product recommendations.

Keywords: Recommender Systems, Collaborative Filtering, Cold Start, Ant Colony Optimization, Singular Value Decomposition, PureSVD, z-score, Item Based Models, top-N Recommendation

ÖZ

DOKTORA TEZİ

ÖNERİ SİSTEMLERİNDE SOĞUK-BAŞLANGIÇ PROBLEMİNE YÖNELİK ALGORİTMA GELİŞTİRİMİ

Hakan YILMAZER

ÇUKUROVA ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ
BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI

Danışman : Prof. Dr. Selma Ayşe ÖZEL
Yıl: 2022, Sayfa: 131
Jüri: : Prof. Dr. Selma Ayşe ÖZEL
: Prof. Dr. Umut ORHAN
: Dr. Öğr. Üye. Alper Kamil DEMİR
: Doç. Dr. Mehmet Uğraş CUMA
Doç. Dr. İrem ERSÖZ KAYA

Soğuk-başlangıç problemleri, öneri sistemlerindeki en önemli zorluklardan birisidir. Bu tez çalışmasında, soğuk-başlangıç problemine 2 farklı yaklaşım ile çözümler geliştirmeyi amaçladık. Farklı senaryolarda kullanılabilecek bir deterministik ve bir sezgisel çalışma yaptık. İlk perspektifte, deterministtik yaklaşımla, Sürekli Karınca Kolonisi Optimizasyonu kullanarak ilk-N öneri listeleri sunmak için ürün tabanlı benzerlik modellerini optimize eden yeni sezgisel bir çatı geliştirdik. Sezgisel yapısı sayesinde, kullanıcılar için kendine has ve her oturuma göre değişen öneri listeleri üretirken aynı zamanda kullanıcının alakası ile listelerdeki ürün çeşitliliği arasında denge kurmayı amaçladık. İkinci perspektifte, deterministik olarak iki yeni işbirlikçi filtreleme tekniğini tanıttık. İlk modelde, örtük verileri kullanarak elde ettiğimiz gram matrisin z-skor normalizasyonuna dayanan ürünler arasında özgün asimetrik bir benzerlik matrisi hesapladık ve ikinci modelde, geliştirdiğimiz ürün benzerlik matrisi ile oluşturduğumuz ürün tahminleri ile seyrekliği azaltmayı amaçladık böylece kullanıcı-ürün matrisinde gizli faktörlerinin daha başarılı ayrışmasını sağladık. Tüm yöntemlerimizi bilinen veri setleri üzerinde test ettik ve yöntemlerimizin, soğuk-başlangıç kullanıcılarda, soğuk-başlangıç sistemlerde ve popüler olmayan ürün önerilerinin sağlanması da dahil olmak üzere çeşitli senaryolarda benzer tavsiye modellerinden daha iyi performans gösterdiğini gözlemledik.

Anahtar Kelimeler: Öneri Sistemleri, İşbirlikçi Filtreleme, Soğuk-Başlangıç, Karınca Kolonisi Optimizasyonu, Tekil Değer Ayrışımı, PureSVD, z-Skor, Ürün Tabanlı Modeller, ilk-N Tavsiye

EXTENDED ABSTRACT

As is known, cold-start is one of the major problems in recommender systems. In the literature, considerable research has been done on this problem. What makes this problem important is that it has a relationship with the solution of many issues in Recommender Systems. In particular, the varying screens and richness of the interaction environments between users and products (Netflix, Spotify, Youtube, Twitch, etc.) have also demonstrated many problems identical to the cold start.

This thesis proposed to develop models for the solution of cold-start problems in various scenarios. Today's modern recommendation systems do not work only through one algorithm. Depending on the case, they could change the models or integrate different models. You may need to follow either a deterministic or heuristic method to establish new links between the user and the products. In this thesis, we have done 2 different studies, one deterministic and one heuristic, which might be used in different scenarios.

In the first perspective, we introduce a new framework that optimizes item-based similarity models to offer top-N recommendation lists by Continuous Ant Colony Optimization, which is a heuristic algorithm. With our new user-specific item-based model, pheromone values are denoted as posterior probabilities of users which are constructed from previous clicks. Our novel model regularizes Lp norms of the clicked items in the selected input similarity model by amortizing them binary cross-entropy and giving stochastic importance to items specific to the user graph which are maximized with hyper-parameter search via in continuous domain.

When comparing the first study with the state-of-art methods in different evaluation scenarios using well-known evaluation metrics and popular datasets (MovieLens, Yahoo, Pinterest), we observed our algorithm offers diverse but relevant and more successful recommendations to the users. The model, which we call AcoRec, provides the opportunity to work with low-dimensional data compared to traditional Ant Colony Optimization models. Thanks to its random heuristic

structure, the most important advantage of our method is its ability to balance high coverage and high recall while producing diverse and session-variate recommendation lists for the users.

In the second perspective, we introduced two novel collaborative filtering techniques for recommendation systems in cases of various cold-start situations and incomplete datasets. The first model establishes an asymmetric weight matrix between items without using item meta-data and eradicates the disadvantages of neighborhood approaches by automatic determination of threshold values. Our first model, z-scoREC, is also regarded as a pure deep-learning model because it performs like a vanilla auto-encoder in transforming column vectors with Z-Score normalization similar to batch normalization. With the second model, ImposeSVD, we aimed to enhance the shortcomings of the PureSVD in cases of cold-start and incomplete data by preserving its straightforward implementation and non-parametric form. The ImposeSVD model relies on the z-scoREC and produces synthetic new predictions for the users by decomposing the latent factors from the imposed matrix.

We evaluated our models on the well-known datasets and found out that our method was outperforming similar approaches in the specific scenarios including recommendations for cold-start users, strength in cold-start systems, and diversification of long-tail item recommendations in lists. Our z-scoREC model also outperformed familiar neighbor-based approaches when operated as a recommender system and gave a closer appearance to the decomposition methods despite its simple and rigid cost framework.

GENİŞLETİLMİŞ ÖZET

Bilindiği gibi, soğuk-başlangıç, öneri sistemlerindeki en büyük sorunlardan biridir. Literatürde bu problem üzerine pek çok araştırma yapılmıştır. Bu sorunu önemli kılan, Öneri Sistemlerinde birçok sorunun çözümü ile ilişkisi olmasıdır. Özellikle kullanıcılar ve ürünler arasındaki etkileşim ortamlarının (Netflix, Spotify, Youtube, Twitch vb.) değişen ekranları ve zenginliği de soğuk başlatmaya benzer birçok sorunu ortaya çıkarmıştır.

Bu tez çalışmasında, farklı senaryolarda soğuk-başlangıç problemlerinin çözümü için modeller geliştirildi. Günümüzün modern öneri sistemleri sadece tek bir algoritma ile çalışmamaktadır. Duruma göre modelleri değiştirebilir veya farklı modelleri entegre edebilirler. Kullanıcı ve ürünler arasında yeni bağlantılar kurmak için deterministik veya sezgisel bir yöntem izlemeniz gerekebilir. Bu tezde, farklı senaryolarda kullanılabilecek bir deterministik ve bir sezgisel olmak üzere 2 farklı çalışma yaptık.

İlk yaklaşımda, sezgisel bir algoritma olan Sürekli Karınca Kolonisi Optimizasyonu tarafından ilk N öneri listeleri sunmak için öğe tabanlı benzerlik modellerini optimize eden yeni bir çerçeve sunuyoruz. Yeni kullanıcıya özel öğe tabanlı modelimiz ile feromon değerleri, kullanıcıların önceki tıklamalardan oluşturulan sonsal olasılıkları olarak ifade edilmektedir. Yeni modelimiz, seçilen girdi benzerlik modelindeki tıklanan öğelerin ikili çapraz entropisini amorti ederek ve sürekli etki alanında hiper parametre araması ile maksimize edilen kullanıcı grafiğine özgü öğelere stokastik önem vererek Lp normlarını düzenler.

İlk çalışmayı, iyi bilinen değerlendirme metrikleri ve popüler veri kümeleri (MovieLens, Yahoo, Pinterest) kullanan farklı değerlendirme senaryolarında en gelişmiş yöntemlerle karşılaştırırken, algoritmamızın kullanıcılara çeşitli ancak ilgili ve daha başarılı öneriler sunduğunu gözlemledik. AcoRec adını verdiğimiz model, geleneksel Karınca Kolonisi Optimizasyon modellerine kıyasla düşük boyutlu verilerle çalışma imkânı sağlıyor. Rastgele sezgisel yapısı sayesinde, yöntemimizin

en önemli avantajı, kullanıcılar için çeşitli ve oturum değişkenli öneri listeleri üretirken yüksek kapsam ve yüksek geri çağırmayı dengeleyebilmesidir.

İkinci perspektifte, çeşitli soğuk-başlangıç durumları ve eksik veri kümeleri durumunda öneri sistemleri için iki yeni ortak filtreleme tekniği sunduk. Birinci model, madde meta verilerini kullanmadan maddeler arasında asimetrik bir ağırlık matrisi oluşturmakta ve eşik değerlerinin otomatik olarak belirlenmesi ile komşuluk yaklaşımlarının dezavantajlarını ortadan kaldırmaktadır. İlk modelimiz, z-scoREC, aynı zamanda, toplu normalleştirmeye benzer z-skor normalizasyonu ile sütun vektörlerini dönüştürmede bir vanilya otomatik kodlayıcı gibi çalıştığından, saf bir derin öğrenme modeli olarak kabul edilir. İkinci model olan ImposeSVD ile, basit uygulamasını ve parametrik olmayan formunu koruyarak, soğuk başlatma ve eksik veri durumlarında PureSVD'nin eksikliklerini iyileştirmeyi amaçladık. ImposeSVD modeli, z-scoREC'e dayanır, uygulanan matristen gizli faktörleri ayrıştırarak kullanıcılar için sentetik yeni tahminler üretir. Modellerimizi iyi bilinen veri kümeleri üzerinde değerlendirdik ve yöntemimizin, soğuk başlatma kullanıcıları için öneriler, soğuk başlatma sistemlerinde güç ve listelerdeki uzun kuyruklu öğe önerilerinin çeşitlendirilmesi dahil olmak üzere belirli senaryolarda benzer yaklaşımlardan daha iyi performans gösterdiğini gördük. z-scoREC modelimiz ayrıca bir öneri sistemi olarak çalıştırıldığında tanıdık komşu tabanlı yaklaşımlardan daha iyi performans gösterdi ve basit ve katı maliyet çerçevesine rağmen ayrıştırma yöntemlerine daha yakın bir görünüm verdi.

ACKNOWLEDGEMENTS

I am grateful to many people for the cooperation, help, and contributions they gave me while preparing this thesis.

First of all, I would like to express my sincere thanks to my supervisor, Prof. Dr. Selma Ayşe ÖZEL, for her support, guidance, and patience throughout my thesis and research. For me, she is an exemplary person and a scientist to be inspired.

I would like to thank my thesis committee members, Prof. Dr. Umut ORHAN, Asst. Prof. Dr. Alper Kamil DEMİR, Assoc. Prof. Dr. İrem ERSÖZ KAYA, and Assoc. Prof. Dr. Mehmet Uğraş CUMA for their contributions and suggestions.

I would like to thank precious people, Prof. Dr. Sera Yeşim AKSAN, Prof. Dr. Mustafa AKSAN, and Asst. Prof. Dr. Umut Ufuk DEMİRHAN who helped me get used to the scholarly knowledge as a result of our research under their guidance as a part of the Turkish National Corpus (TNC) team.

I would like to thank to my dear friends Özgül Berber YAĞDIRAN, Uğur YAĞDIRAN, Asst. Prof. Dr. Jale BEKTAŞ, and my colleague Yasin BEKTAŞ whose support I have always felt during my Ph.D.

I would like to express my heartfelt thanks to my mother Şenel YILMAZER, who always wanted me to be a 'Doctor', for her eternal love and support. I would also like to express my heartfelt respect and thanks to my father Yusuf YILMAZER who, even at the age of 43, always saw me as a 10-year-old pupil and always asked when my Ph.D. would be finished. The sacrifices he made for my education are special and they guide me in my daughter's education life today.

I would also like to thank my brothers and sister, father-in-law, and mother-in-law.

I would like to thank my sweet daughter Ekin YILMAZER for her great patience and love. During my thesis, she always understood the times that we could not be together, and she became my biggest source of motivation. She was a baby

when I started my Ph.D. and she turned out to be a lady who combed my hair and encouraged me on the day of my defense.

Last but by no means least, I would like to express my deepest love to my dear wife Dr. Meryem ÖZDEMİR YILMAZER. Her presence always guided and empowered me during my thesis. She was my source of motivation even when the times I was most depressed. She was always patient on this long road. She also contributed a lot with her knowledge and experience in every stage of my Ph.D. studies.

TABLE OF CONTENTS **PAGE**

LIST OF TABLES

LIST OF FIGURES

ABBREVIATIONS

ACO : Ant Colony Optimization
ACO_R : Continuous Ant Colony Optimization
ALS : Alternating Least Squares
BCE : Binary-Cross-Entropy
BX : The BookCrossing
CBF : Content-Based Filtering
CDAE : Collaborative Denoising Auto Encoder
CF : Collaborative Filtering Computing Center
COS : Cosine
CSV : Comma Separated Value(s)
DCG : Discounted Cumulative Gain
Diag : Diagonal
Dotp : Dot Product
DT-BAR : Dynamic T-BAR
$EASE^R$: Embarrassingly Shallow Auto Encoder
FISM : Factored Item Similarity Model
FKA : Formally Known As
HOSLIM : Higher-order Sparse Linear Method
HR : Hit Rank / Recall
ICF : Item-Based Collaborative Filtering
IDCG : Ideal Discounted Cumulative Gain
JAC : Jaccard
k-NN : k-nearest-neighbor
L_1*-norm* : Sum of the magnitudes of the vector
L_2*-norm* : Sum of the magnitudes of the vector
LORSLIM : Low-rank Sparse Linear Method
L_P*-norm* : p valued norm of the vector

ML-10M : MovieLens-10M
ML-1M : MovieLens-1M
nDCG : normalized Discounted Cumulative Gain
NP-hard : Non-deterministic polynomial-time hard
pdf : Probability Density Function
RS : Recommender System
SGD : Stochastic Gradient Descent
SLIM : Sparse Linear Method
STARS : Semantic-enhanced Trust-based Ant Recommender System
SVD : Singular Value Decomposition
TARS : Trust-based Ant Recommender System
TCFACO : Trust-aware collaborative filtering method based on ant colony optimization
top-N : a list of N items
TRUBA : TUBITAK ULAKBIM, High Performance and Grid
TSP : Travelling Salesman Problem

1. INTRODUCTION

Recommender Systems (RSs) are collection of information retrieval, data mining, and machine learning tools aimed at predicting and recommending the new users and items (such as movies, books, music, online products, TV shows, and websites) to propose a liking from predominantly large data. Initial work on recommender systems began in the mid-1990s (Adomavicius & Tuzhilin, 2005). With the increasing number of websites, and widespread use of e-commerce and social networking sites, RSs have recently become an important field of Intelligent Systems that have been dealing with the increasing extent of social networks, e-commerce sites (or applications), and entertainment media services.

1.1. An Overview of the Recommender Systems and Taxonomy

Recommender Systems (RS) are the unity of studies conducted in the field of presenting information to the users/customers via filtering attractive information for them. RSs aim to provide the requested information by using different filtering methods or combining some of the giant data existing in the field of the Internet and the market. RSs can obtain the data either from explicit sources such as user ratings, friendships, and relations, or implicit sources such as user likes, user logs, habits, or clicks.

RSs generally work like as exampled in Figure 1.1.; is to recommend similar items to the items that a user attracted, or to find similar users to that user and recommend item(s) that those similar people attracted. Here the word 'attraction' could mean different meanings in different domains.

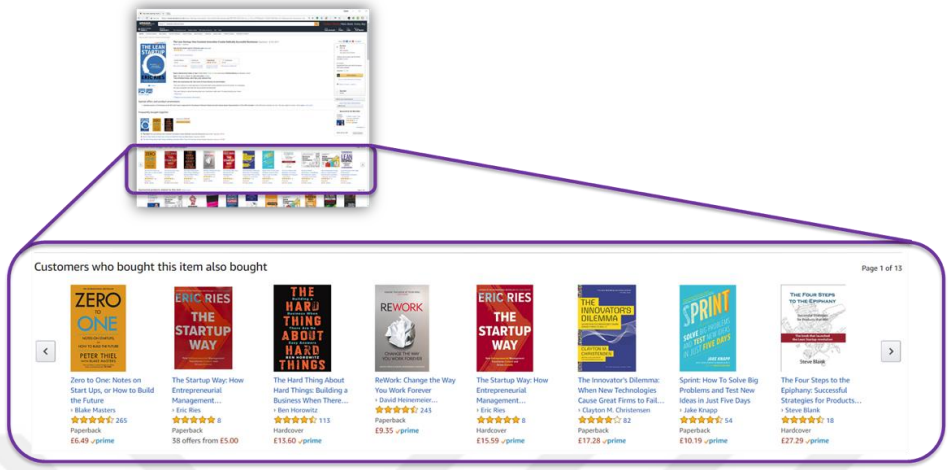

Figure 1.1. An example screen from an item (Amazon, 2022)

For example, in the movie/TV domain, users watch the items. When the product is a book, users can read the book, listen to the song when domain is music, or some products can be eaten, visited web pages, added to the cart, or purchased. In this thesis, we generalize and name all these attractions as 'clicked'. Users can give feedback to the system by liking, rating, or commenting on products.

As explained in Figure 1.2. the recommendation systems convert the input from different fields and data sources into output with a function.

In the last quarter-century (especially during the COVID-19 pandemic), research in the area of Recommender Systems started to gain more importance with the inclusion of Youtube, Netflix, Instagram, Twitter, Spotify, and many similar web services / social media sharing sites in our lives.

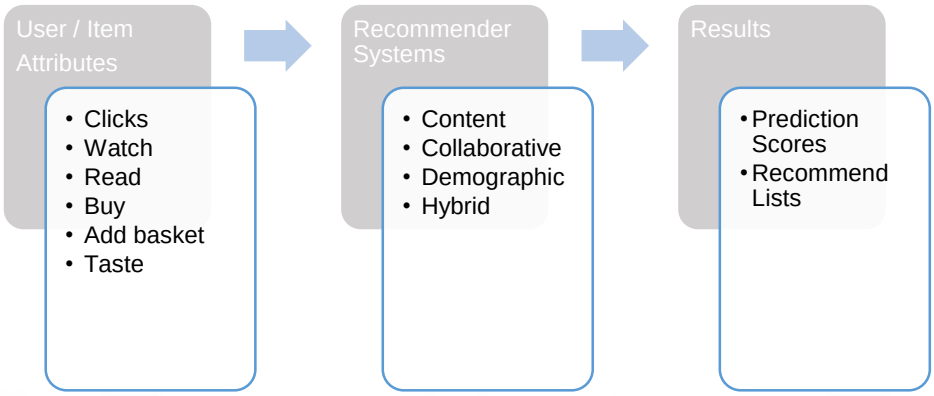

Figure 1.2. Recommender System in a function view

These studies include generating high-quality recommendations, performing many recommendations for millions of users and items, providing high coverage against data sparsity, quickly adapting new users to the system with satisfaction, scaling issues, localizations, etc. (Sarwar et.al., 2001).

Recommendation algorithms can be classified according to various conditions. However, the most common classifying found in the literature refers to how they use the information of user preferences for items, for which four categories are commonly established as shown in Figure 1.3 (Resnick et.al., 1994; Adomavicius & Tuzhilin, 2005; Pazzani, 1999).

- Content-based filtering; recommendations are made from similar products as content.
- Demographic filtering; recommendations are presented according to clusters of users/items with common characteristics (age, gender, location, etc.)
- Collaborative-based filtering; recommendations are formed from users with similar tastes.
- Hybrid filtering; combines at least two filtering methods explained above.

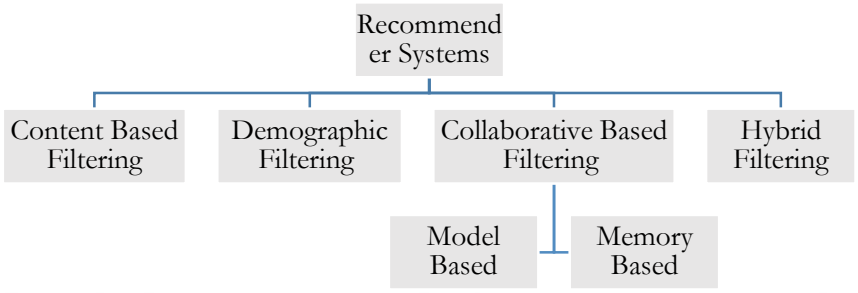

Figure 1.3. Recommender Systems categories

1.1.1. Content-Based Filtering (CBF)

Content-based methods build user profiles based on the features and descriptions of the items rated by the user and do not use other users' preferences for issuing recommendations (e.g., in a movie domain, if the user watched some comedy movie in her history, the filtering method will probably recommend a recent comedy movie that she has not yet watched). One of the advantages of content-based methods is that they can deal seamlessly with the new item problem that is they are able to recommend new items for which there is no user feedback, as opposed to collaborative filtering algorithms.

Content-based algorithms, however, are very dependent on the recommendation domain, which contrasts with the generality of collaborative filtering methods. Additionally, one of the major problems is that content-based approaches may suffer from over-specialization which is, that they have a natural tendency to recommend the same items that are before recommended to that user, and users may be bored (Resnick et.al., 1994).

1.1.2. Demographic-Based Filtering (Knowledge-Based Filtering)

In RSs, demographic information is used to group users according to a familiar class. The demographic attribute may vary depending on the domain in which the recommendation system operates. In general, users can be grouped according to features such as age, gender, occupation, location, education, budget, mood, and similar recommendations can be offered to users in the common group (Pazzani, 1999).

1.1.3. Collaborative Filtering (CF)

Collaborative Filtering describes the family of algorithms that exploit the users' consumption patterns of the items in the recommendation domain, without making use of any domain-specific characteristics of the items, such as their content or categorization.

The main advantage of this type of algorithm is its independence concerning the recommendation domain in which they are applied. They have been claimed to be more effective than other approaches, such as Content-based algorithms. Collaborative Filtering algorithms can be classified into two types:

Memory-based; Memory-based methods are characterized by their simplicity since a minimal or no learning phase is involved. This lack of learning phase provides several advantages, such as easiness of implementation, immediate incorporation of new data, and comprehensibility of results. Memory-based methods, however, may suffer from scalability issues and a lack of sensitivity to sparse data.

Model-based; The recommendations are based on a model that is previously learned from the user data. Model-based methods take a different approach to exploiting collaborative filtering data. The algorithms of this family depend on a learning phase, in which a descriptive model of user preferences based on the observed data is built to make predictions. These methods are inspired by machine learning techniques such as Neural Networks, Bayesian networks, Clustering, Fuzzy Systems, Genetic Algorithms, Singular Value

Decomposition (SVD), Latent Semantic Analysis, Bee Colony, and Ant Colony Optimization (ACO) among others (Bobadilla et.al., 2013).

1.1.4. Hybrid Filtering

Hybrid models are one of the most widely used filtering methods. Hybrid methods have been proposed to avoid the limitations of collaborative filtering and content-based algorithms when used separately (Balabanović & Shoham, 1997; Burke, 2002; Adomavicius & Tuzhilin, 2005).

Mixed models combine different recommendation models to perform a more useful recommendation quality, merging the advantages of models that included in hybrid structure.

There are three basic strategies in hybrid recommendation: The first strategy combines the final recommendation results produced by two or more recommendation algorithms. The second strategy utilizes a recommendation algorithm as a framework and includes other algorithms onto it. The third strategy incorporates various models into a cooperative recommendation model and then produce recommendations.

1.2. Limitations of Recommender Systems

Researchers on Recommender Systems study on many different problems. These studies are usually about performance, error and satisfaction (Bobadilla et al. 2013). Some problems are related to each other and the solution is also a guide for other problems. The followings are the major problems studied in the literature.

1.2.1. Data Sparsity

One of the major problems of recommender systems is how they can solve the problem of data sparsity. The lack of relationships between users and

items in recommendation systems with large quantities of data makes it impossible to make sufficient measurements for collaborative filtering calculations (Adomavicius & Tuzhilin, 2005).

Especially, CF algorithms are inadequate in cases where the data is sparse. This problem, which researchers mostly studies on, is one of the reasons for other challenges in recommendation systems.

1.2.2. Scalability

The data are often large and scattered (sparse), with large sites containing millions of users and items. It is very important to look for recommendation algorithms that facilitate and parallelize (or both) the computational cost considerations (Sarwar et.al., 2002).

1.2.3. Cold-start

One of the primary limitations that must be overcome in RS is providing recommendations in the case of cold-start states where there is no data or only a limited amount of data about the user or the item. In such cases, RS cannot provide effective recommendations (Sarwar et.al., 2011).

There are three observed types of the cold-start problems; ‘new item’, ‘new user’, or for both ‘new system’. In the ‘new item’ problem, it is hard to recommend the new item for a user because the new item has been recently added and has a very limited amount of meta-data. In the ‘new user’ problem, where there is no data about the user, RS could not draw on relations about newly registered users or users who do not have many collaborations on the system; as a result, the system could be inadequate for developing the links between users and items because the evaluated data cannot provide information about the users and the items. In the ‘new system’ case where there is no data about both the users and the items, systems could be inadequate for developing the links between users and items.

1.2.4. Novelty-Diversity

The ‘diversity’ and ‘novelty’ of the recommendations offered to users are the other issues to be dealt with (Adomavicius & Tuzhilin, 2005; Hurley & Zhang, 2011; Bobadilla et.al. 2013). For instance, the recommendation of popular items may not be valuable to users as these items are already familiar to users and they could be bored. CF methods based on the analysis of cooperative behaviors between users might be inadequate to solve this problem because they overshadow unpopular connections and cause them to be ignored as a result of their tendency to offer collaborations deeper among popular users and favored items because of their frequencies. Whereas, the recommendation of unpopular items has always been more attractive to the users (Yin et.al., 2012; Anderson 2006).

1.2.5. Evaluation

The success of the solution to the problem to which a designed RS model is adapted is measured by the correct evaluation strategy. Evaluation is also an important tool in choosing the right model for different scenarios in a commercial system using many models. Even a very small difference between models makes a difference in satisfaction level in systems with millions of users. Evaluation is one of the difficult tasks in RSs. Since it is difficult to evaluate very large data, sampling methods and choosing the right metrics according to the scenarios are important factors.

1.2.6. Real-time recommendations

For a recommendation system, offline approaches would be better for evaluating new models and posterior predictions for users or when the data do not change significantly over time. However, in recommendation systems, data is approached in real-time and it is necessary to provide instant recommendations to users. Providing recommendations live on such large datasets is a difficult task.

1.2.7. Over-Specialization

Recommender systems mostly offer recommendation lists for each user based on their history on the system, which might turn out to be similar, uncompelling, and poor-quality recommendations for the users (Balabanović & Shoham, 1997; Ar & Bostanci, 2016; Olaleke et.al., 2021). This challenge forces us to solve the problem which is called the over-specialization problem.

1.2.8. Ethic Problems

RS can obtain this data either from data like user ratings, friendship, relations which exist explicitly or from implicit data which is not shown to end user like user logs and user habits.

Users' information in the system is legally private, and some users may not allow the recommendation system algorithm to use this information. In this case, only legally obtained information can be used and this information may not be sufficient in some cases and it may be difficult for the system to give personalized recommendations to users.

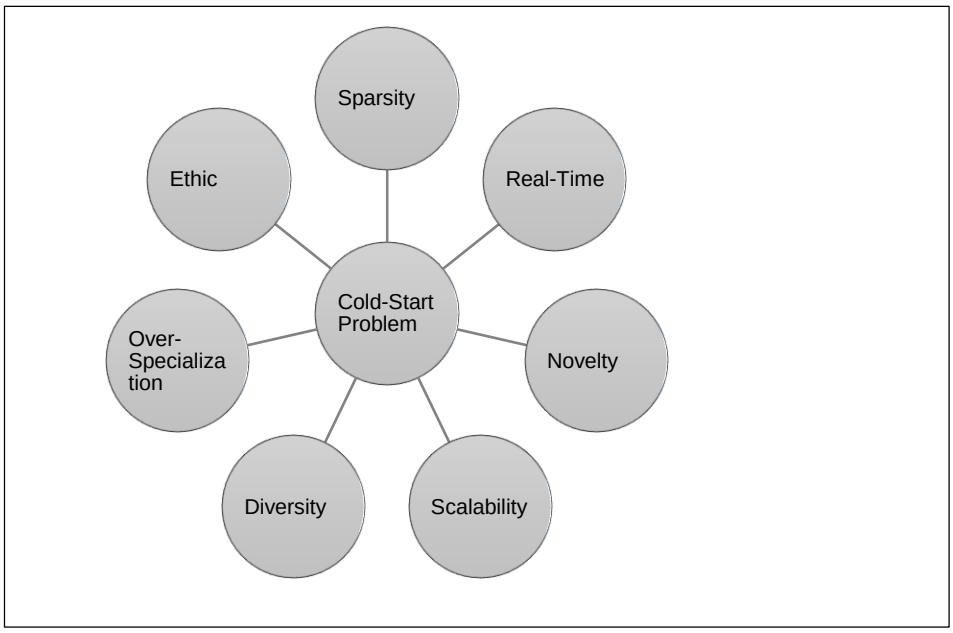

Figure 1.4 Cold-start and other problems in recommender-systems

1.3. Relation among other challenges and cold-start

We have described in the previous section that one of the problems encountered in recommender systems is the cold-start problem. As shown in Figure 1.4, with this thesis, we believe that the answer to most of the mentioned problems in recommender systems is the key to the cold-start problem. Therefore, the strategies to solve the cold-start problem become a solution for other challenges.

The challenge to ensuring the quality is the presence of cold-start users. Although most recommender systems approach the problem with cold-start users in offline settings, it is necessary to follow their tastes simultaneously on the system. All users should be considered cold-start users because of their ever-changing and unexpected habits. However, recommender systems mostly offer recommendation sets for each user based on their history on the system, which might turn out to be similar, uncompelling, and poor-quality recommendations for the users (Balabanović & Shoham, 1997; Ar & Bostanci, 2016; Olaleke et.al., 2021). This challenge forces us to solve another problem with related cold-start users which is called the over-specialization problem.

While most systems approach the cold start problem, they consider the data from zero time, whereas for the accuracy of the analysis, it is necessary to look from an unknown starting point and see the system in action. In a personalized recommendation model, different recommendation lists are created for each user base on users' past tastes. After a particular time, RS resumes to recommend the same lists or is biased to popular items (Olaleke et.al., 2021). However, most users prefer diverse item recommendations on their screens as shown in Figure 1.5. To satisfy such users to keep from over-specialization problems, solutions force us to solve cold-start users.

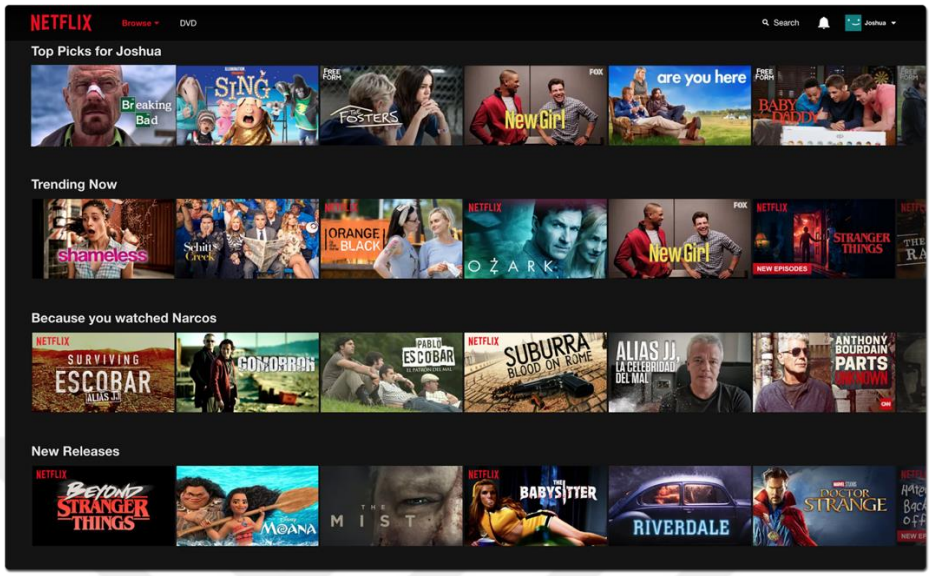

Figure 1.5. A screenshot from Netflix Application has a four-row recommendation list for a user

Recommendation of unpopular items includes strong diversity and novelty since they have not been in contact with many users or related to other items. In sparse datasets, it becomes difficult to make accurate recommendations. The systems provide recommendations based on heat users; therefore, users who have just logged into the system (cold-start users) or have different tastes might be omitted. And the solution of this problem also redirect us to cold-start problem.

In the ethical problem, we mentioned that we might not have information about users in some cases. In such a situation, even if the user is a hot user using the system actively and does not allow the system to use her information, she should be considered a cold-start user.

We explained the importance of providing live recommendations in recommendation systems in the real-time recommendations problem (See 1.2.6). In such cases, we need to abstract recommends based on users' current sessions or recent-short histories, and it is necessary to look at the instant habits, not their past interactions. Such users can also be seen as cold-start users due to the scarcity of data.

In a real scenario, social media systems have a giant interaction graph due to the number of users and items. It takes a long time to perform traditional calculations on a large graph. Considering the whole product set on the system that attracts the consideration of users does not represent even 1% of the total product scale, such users can naturally be seen as a cold-start user for a recommender system.

The vital processes of a recommender system are to increase the connected nodes of the user-item graph and produce more accurate predictions between users and new items. While doing this; however, the system must find user-specific relations, which is considered the quality. Data sparsity and cold start essentially point us towards the same solution, enrichment of the user-item graph.

The data sparsity problem (indirectly the cold-start problem) has led researchers mainly to two areas of study: dimension reduction and graph-based techniques. Latent factors were tried to be extracted from the non-complete user-product matrix with dimension reduction techniques. Graph-based techniques, on the other hand, endeavored to find connections between unconnected users and products (Ricci et.al., 2015).

1.4. Purpose of thesis

Recently, visual media platforms (YouTube, Spotify, Netflix, Twitch, etc.) have been increasingly used particularly during the Covid-19 lockdown periods. Media services around the world tend to offer recommendation lists to the users on their phones, tablets, or television screens according to the item groups they have tasted. These groups of recommendations based on the user's previous likes, their history on the platform, trending items, or demographically related items are presented in horizontal or vertical forms on the main screens of many media platforms (see an example Figure 1.5). As a result of the developing and competing recommender system technologies, users expect personalized or session-based recommendations on the platforms (Hidasi et.al., 2015). However,

generating online recommendations in live recommendation systems is a challenging task due to the absence of the initial or accomplished state of the data, which requires the evaluation of ongoing and noisy-data systems rather than employing data from the scratch. Although the recommender systems tend to use traditional deterministic algorithms (CF and CBF) offering the same recommendations for all users (Olaleke et.al., 2021), they need to provide constant updating and diversification of the home screen recommendations because of the changing tastes of the users. As a result, researchers in the field of recommendation systems have recently considered Heuristic Methods and Deep Learning Methods to offer continuous and variable recommendations (Vargas, 2015). These methods might be successful as they not only offer recommendations to cold-start users but also volatile encouragement to existing users. In addition, these methods might be considered as fast, robust, and parallelable alternatives as they can deal with the costs of these personalized tasks that are performed for a tremendous number of users in the systems.

This thesis aims to design new algorithms with the solution to the cold-start problem. The concern of this study is to handle cold-start problem together with others by paying attention to mutual solutions in the algorithms designed (See Sections 3.1 and 3.2). Since cold-start user and cold-start system (sparsity) are considered to be relatively more difficult problems compared to the cold-start item problem, these two are particularly emphasized in this thesis. This study aims to find out solutions particularly to the cold-start problem while indirectly overcoming many recommender system problems.

The cold-start problem is a field of study that attracts attention, especially in the industrial and academic fields. Therefore, this study will add values to the literature.

The contributions of this thesis to the literature could be summarized as follows;

- This thesis handled more than one problem by associating the cold-start problem with other problems.
- The models were developed in various domains and designed to be industrial across the board.
- These models are designed to be easy to implement, appropriate for parallelization, parametrically easy, and forthcoming to evolution for researchers.
- New perspectives were brought to the studies in the field.
- The designed models keep up with existing and transforming technology.

1.5. Outline of thesis

The remainder of the thesis is organized as the following:

Chapter 2 gives a detailed literature survey on the evolution of the recommender systems, common solutions about cold-start problem, related works about SVD-based and ACO-based studies in recommender systems, top-N recommendations systems survey.

Chapter 3 explains our proposed models in two sections. First Section introduce AcoRec which is our novel model about cold-start and related problems. Section 2 of this chapter introduce z-scoREC and ImposeSVD which are our other novel models rely on regression and dimension reduction to solve cold-start and other related problems.

Chapter 4 presents and discusses the experiments in detail all of three proposed model; and compares results of our models with state-of-art studies in the literature.

Chapter 5 summarizes our conclusions, primary contributions of this thesis; and gives the future purposes and expectations of the work.

2. RELATED WORKS

2.1. Evolution

Research on RSs started to increase during the 1990s. The studies of Malone et al. (1987) and Resnick et al. (1994) found out filtering types in the RS field, which were determined as Content-Based Filtering (FKA Cognitive filtering) and Collaborative Filtering (FKA Social filtering). Collaborative Filtering (CF) term was first used in the literature in 1992 in Tapestry's recommendation system (Goldberg et.al., 1992). Later, Shardanand and Maes (1995) developed *Ringo*. CF needs consumption patterns of users or items without considering the domain properties. Content-Based Filtering (CBF), on the other hand, needs meta-information about users or items, which varies depending on the domains. Balabanovic and Shoham (1997) introduced the first hybrid application by combining two filtering techniques and gave a new direction to research in the RS field. To provide more efficient results for hybrid studies, Basu et al. (1998) developed the Ripper algorithm by creating bot users. Breese et al. (2013) categorized RS into memory-based and model-based algorithms. The review of Herlocker et al. (2004) about how to evaluate RSs is one of the most prominent studies in the literature. Adomavicius and Tuzhilin (2005) reviewed numerous studies shown until 2005, where they underlined how to increase the capabilities of RSs by revealing common limitations including the cold-start problem. Bobadilla et al. (2013) published a comprehensive review of the RS studies in the literature.

2.2. Cold-Start Studies in Recommender Systems

When relationships between nodes (i.e., users, items) are missing or inadequate in the dataset, establishing new relationships is an important challenge for RSs. Therefore, a growing number of studies in the literature have been searching out the ways to build new relationships between nodes in the case of limited data by applying CBF or CF models.

To illustrate, Kim et al. (2010) employed collaborative tagging as an approach to collect users' preferences and tastes. Basilico and Hofmann (2004), on the other hand, developed a framework that incorporates all available information by using a suitable kernel or similarity function between user-item pairs.

Weng et al. (2008) combined the implicit relations between users' preferences for items with additional taxonomic preferences to make better quality recommendations to alleviate the cold-start problem. In addition, Loh et al. (2009) represented users' profiles with information extracted from their scientific publications.

Other than the sole use of CBF or CF models, hybrid models have been employed by others to overcome cold-start problems resulting from the sparsity of the dataset or unavailability of data (Basilico & Hoffman, 2004; Kim et.al., 2010; Pazzani, 1999).

Pazzani (1999), proposed a hybrid framework that merges different algorithms: CF, CBF, and Demographic Based Methods.

Jamali and Ester (2010) relied on trust between users on a trusted network instead of user similarity to deal with data sparsity and the cold-start problem.

Massa and Avesani (2009) used explicit trust as input, along with a user-item rating matrix to predict ratings.

Bobadilla et al. (2010), Chandelier et al. (2008) and Ahn (2008) recommended new collaborative filtering metrics that improve RSs.

Among others, graph-based approaches (Ning et.al., 2015), a combination of content and collaborative filtering (Schein et.al., 2002), collection of clickstreams from user experiences (Embarak, 2011), and pairwise regression-based models (Park & Chu, 2009) were utilized to overcome the cold-start problem. Fouss et al. (2006) showed uses of the graph-based kernel methods in CF.

Son (2016) has summarized papers based on cold-start and categorized the cold-start based studies into four sections. He also evaluated these categorized

methods. Guo (2013) formed a new rank profile for the active users by merging the ratings of trusted neighbors. He then assessed this approach through a Bayesian similarity measure, which considers both the direction and length of rating profiles.

Regarding the quality of the recommendations, Herlocker et al. (2004) has pointed out many criteria according to which diversity and personalized user satisfaction are more important factors for the recommendation of unpopular items.

2.3. Ant Colony Based Studies in Recommender Systems

In literature, the studies conducted on the use of agents in recommendation systems with cold start problems, Good et al. (1999), Park et al. (2006) and Sarwar et al. (2008) tried to produce solutions for Cold-start problems by using bot filters.

Sarwar et al. (2006) created rating bots and these bots rated new documents in the newspaper. With this new rating, they qualified and classified new documents and represent them to their users. They aimed to decrease gaps in the recommendation matrix and spread out the sparsity. They used 'Usenet news Article' dataset. In their study, they created artificial rating bots, and these bots rated recently added documents considering different criteria. With the help of these rates given by bots compromised by artificial users, they tried to enable the integration of recently added documents to the recommendation system and minimize the infrequency in connection.

Good et al. (1999) produced various bots with different characteristics and unified them with both CBF and CF algorithms. They suggested that more successful results would be obtained if ratings that bots give for films were unified with user ratings and calculated in the user-item matrix.

Park et al. (2006) also worked on filter-bots for cold start problems. They created seven basic filter bots the resolve the problem and used these ratings in user-based and item-based estimations in CF.

There are many ACO studies based on RS in the literature. Sobecki and Tomczak (2010) used real data for recommending student courses based on ACO. T-BAR is a probabilistic model on the ACO algorithm, which is considered to develop the efficiency and coverage of predictions for users (Bellaachia & Alathel, 2012). However, this algorithm suffers from its failure to deal with cold-start users. The authors proposed an update DT-BAR (Dynamic T-BAR) to overcome this problem (Bellaachia & Alathel, 2014). Bellaachia et al. (2016) introduced ALT-BAR with averaged localized trust-based ant recommender for cold-start recommendations. Massa and Avesani (2009) proposed Mole-Trust is a basic CF algorithm that uses the Pearson Similarity and Trust in recommender systems.

Bedi and Sharma (2012) introduced the Trust-based Ant Recommender System (TARS) to produce recommendations by merging the assumption of trust between users and taking the best similar users based on the ACO. During iterations, TARS generates new relationships between users and produces predictions with the help of new, updated trusted users.

Semantic-enhanced Trust-based Ant Recommender System (STARS) introduced a more progressive model that tried to eliminate the disadvantages of the TARS model and included semantically user similarity with clusters (Gohari et.al., 2017).

TCFACO has also studied trust statements between users and developed an ACO-based CF method for effectiveness predictions for users (Parvin et.al., 2019).

Tengkiattrakul et al. integrated SVD-based user factors and trustworthiness for user-similarity on ACO (Tengkiattrakul et.al., 2016; Tengkiattrakul et.al., 2018). Kaleroun and Batra (2014) upgraded TARS with item deviation distance products in the prediction formula and evaluated for the Shilling Attack, Cold-start Users, Sparse Matrix, and Grey Sheep Users problems.

Liao et al. computed user pheromones and item pheromones separately and combined them in rating prediction to produce ranking lists (Liao et.al., 2020a; Liao et.al., 2020b).

Nadi et al. (2011) used a fuzzy-based ant colony system for website recommendation, they used Jaccard-based user similarity and they fuzzified the user-item interaction matrix.

Detailed information about the algorithms that develop the recommendation system with the ant colony was given in Appendix C.

The typical approach in these ACO-based studies is as follows;

• Calculation of user similarities (e.g., Cosine, Jaccard, Pearson, Trust measures)

• Obtaining users as nodes and selection of similar users with ACO

• Analyzing the new recommendations from similar neighbors (users) from Resnick's prediction formula (Resnick et.al., 1994).

Different from the studies above, our ACO algorithm is item-based. In our study, the nodes represent the items in the ACO graph structure, and the edge values of the items show the importance that reflects the likelihood values of the user to the relevant item. The other distinction is that we tried to find a heuristic on popular items in the continuous domain with auto hyper-parameter tuning.

2.4. SVD Based Studies in Recommender Systems

SVD-based methods apply the process of smoothing the rating matrix by reducing the original matrix size to the low-ranks. However, incomplete matrices cannot be decomposed by SVD-based methods. Therefore, researchers applied matrix factorization algorithms over non-null data using *Stochastic Gradient Descent* (SGD) (Robbins & Monro, 1951) or *Alternating Least Squares* (ALS) methods (Zhou et.al., 2008). Funk (2006) used a simple linear regression model to calculate user and item factors for estimating rating predictions. Paterek (2007)

composed an advanced model by adding user and item biases. Koren has contributed with new models such as *SVD++*, *timeSVD++* where he added *k-nearest-neighbor* or *time factors* to earlier developed models using (SGD) or (ALS) (Koren, 2008; Koren et.al., 2009; Koren, 2009). But all these methods are not real SVD-based methods that use algebraic calculus for decomposition.

The first example of using the original decomposition method via SVD in the CF field is the work of (Sarwar et.al., 2000). Cremonesi et al. (2010) introduced the PureSVD, which bases on an estimation of the low-rank latent factors from the rating matrix by SVD followed by imputing null values with zero on the rating matrix. EigenREC demonstrated that PureSVD's prediction formula actually only needs item factors, and this can be calculated more easily with both *Eigenvalue Decomposition* and *Golub-Kahan-Lanczos Bidiagonalization* (Nikolakopoulos et.al., 2017; Nikolakopoulos et.al., 2019). In addition to these, HybridSVD has claimed that PureSVD can be strengthened with side information in cases where the CF is inadequate. They added item features to the rating matrix by effectively using *Cholesky decomposition*, and then this new auxiliary matrix was performed for low latent factors in SVD (Frolov & Oseledets, 2019).

In another study, Ghazanfar et al. Ghazanfar and Prugel (2013) studied how to carefully increment the rating matrix before the SVD process by developing a model they call ImputedSVD. In their study, they performed improvements on cold-start, long-tail, and sparsity issues by applying the art-of-state methods in the literature as imputation methods. Although the study of Ghazanfar and Prugel (2013) bears some similarities with our study in terms of providing improvements on SVD-based models to overcome the most common problems in RSs, the methods of our research are different in many aspects. First of all, (a) while the ImputedSVD adopted the well-known methods in the literature as the imputation method, we introduced a novel item-similarity matrix in our research and generated a less parametric imputation method, which is suitable for many domains. (b) Based on the simplicity of the PureSVD, we needed item vectors only that are taken from decomposition to estimate

prediction scores. ImputedSVD; however, carried user factors of SVD for prediction, which is a costly method in terms of handling time. (c) While ImputedSVD works on rating prediction, our research is a ranking-based study for generating top-N recommendation lists. Finally, (d) the offline cost of our method is low and there is no need for calculations for the cold-start users in the online process.

2.5. Studies on top-N Recommendation

RSs methods are divided into the two corresponding models, which are 'rating-based', and 'ranked-based' models to measure the prediction scores. Rating-based models predict the user scores based on their unrated items by normalizing the real rating range that the user would give to the item. On the contrary, ranking-based models predict a list of *N* items ***(top-N)*** that the users may like. Therefore, ranking-based models do not need to scale real range values (e.g. min 1- max 5) to provide flexibility while developing algorithms.

To our best knowledge, Karypis (2001) did the first study on the ranking-based model to provide *top-N* recommendation lists. In another study, Deshpande and Karypis (2004) employed the *k-nearest-neighbor* (*k-NN*) estimation as a ranking-based model to provide top-*N* recommendation lists to the users. Cremonesi et al. (2010), on the other hand, proposed evaluation methods of *top-N* recommendation systems and introduced the PureSVD model to measure the diversification of long-tail items in recommendations. Hurley and Zhang (2011) showed how novelty and diversity values of *top-N* recommendation lists could be improved. Rendle et al. (2012) introduced an optimization approach by learning over click pairwise of users on items. Ning and Karypis (2011) developed SLIM where he used linear regression to construct a coefficiency matrix with L_1*-norm* and L_2*-norm* regularizations and used a coefficiency matrix for offering recommendation lists. In a later work, higher-order item relationships for SLIM were added by HOSLIM developed by Christakopoulou and Karypis (2014). Besides, in another method called LORSLIM, Cheng et al. (2014) performed low-

rank optimization on the SLIM's coefficiency matrix. In another study, Kang et al. (2016) aimed to complete the rating matrix with a non-convex optimization problem. Kabbur et al. (2013) developed FISM to produce the top-N recommendation lists by training the two item factors with the help of the loss function. Cooper et al. (2014) introduced a simple graph-based algorithm $P^3\alpha$ that implements three steps *random-walk* between users and items. In a later work (Christoffel et.al., 2015), the authors upgraded the previous graph-based algorithm and created $RP^3ß$ by adding an item-popularity parameter to develop the success of long-tail item recommendations. Nikolakopoulos and Karypis (2019) introduced RecWalk, which use the power of *random-walk-based* methods to capture new rich network interactions. Nikolakopoulos et al. (2019) introduced the PerDif as an implementation of *diffusions* over item-item graphs for live personalized user recommendations. Both RecWalk and PerDif approaches gave good results with the assist of item-based models such as Cosine and SLIM.

In recent years, auto-encoders models have shown good results on recommendations. For example, CDAE developed by Wu et al. (2016) used auto-encoders with neural networks for an item-based *top-N* recommendation algorithm. Liang et al. (2018) extended CDAE to multinomial likelihood instead of Gaussian likelihood and they used variational auto-encoders on implicit feedback. Shenbin et al. (2020) introduced a new auto-encoders structure that outperformed the previous auto-encoder structured recommendation models. Chen et al. (2018) proposed to merge user ratings and side-information by using a variational auto-encoder structure to produce recommendation lists. Steck (2019) introduced $EASE^R$ as a simple linear and vanilla auto-encoder model that outperforms the state-of-the-art collaborative filtering approaches for huge sparse data.

3. MATERIALS AND METHODS

In this thesis, three original models have been developed from two different perspectives one of which is deterministic and the other one is non-deterministic. All models were evaluated over various cold start scenarios on well-known datasets.

In the first perspective, a new recommendation model with ACO, an intuitive method in the category of graph-based techniques was developed.

From a view of first perspective, we introduce a new framework that optimizes item-based similarity models to offer top-N recommendation lists by Continuous Ant Colony Optimization, which is a heuristic algorithm. With our new user-specific item-based model, pheromone values are denoted as posterior probabilities of users which are constructed from previous clicks. Our novel model regularizes *Lp* norms of the users' clicked items in the selected input similarity model by amortizing them *binary cross-entropy* and giving stochastic importance to items specific to the user-graph which are maximized with hyper-parameter search via in continuous domain.

In the second perspective, we introduced two novel collaborative filtering techniques for recommendation systems in cases of various cold-start situations and incomplete datasets. In this perspective, the first model establishes an asymmetric weight matrix between items without using item meta-data and eradicates the disadvantages of neighborhood approaches by automatic determination of threshold values. This model, z-scoREC, is also regarded as a pure deep-learning model because it performs like a vanilla auto-encoder in transforming column vectors with Z-Score normalization similar to batch normalization. With the second model of this perspective, which we called ImposeSVD, we aimed to enhance the shortcomings of the PureSVD in cases of cold-start and incomplete data by preserving its straightforward implementation and non-parametric form. The ImposeSVD model relies on the z-scoREC, produces synthetic new predictions for the users by decomposing the latent factors from the imposed matrix.

3.1. A Non-deterministic Perspective, AcoRec

3.1.1. NP-Hard Problems and Deterministic Recommendation Models

Deterministic recommendation models are robust algorithms despite their simple structures. For instance, neighborhood models or regression models can be overwhelmed by many algorithms (Sarwar et.al., 2001; Dacrema et.al. 2019). In deterministic recommendation models, users are given a set of recommendations *{S}* at time *t1*, and this set *{S}* remains the same as long as there is no change in the model between time *t1* and *t2*. However, we might not be foolproof that the results of an algorithm that produces deterministic solutions with discrete parameters are precise and recall results are accurate, but we could acknowledge them as thriving or sufficient based on the evaluation results (Olaleke et.al. 2021). Many researchers obtain evaluation results in algorithms by averaging all the results of experiments, hence these results could vary depending on the selection of dataset, sampling of these datasets, selected metrics, and hyper-parameters evaluations (Herlocker et.al. 2004; Dacrema et.al. 2019).

In heuristics, the recommendation set *{S}* can suggest different sets *{S}* without changing the model because of the randomness of its core and this could be an attractive situation for users. But there is a challenge in providing various *{S}* sets for a recent user. We know in recommendation systems that the recommendations are given to a user *u* are never certain, so what is being done in this study is an inferential estimate, just like a top-N recommendation list. As recommendation systems are based on predicting the items that users would like and do not provide definitive results, the recommendation process is an NP-hard problem (Hammar et.al., 2013; Vahabi et al., 2015; Nembrini et al., 2021). According to the feedback got, the predictions are updated from time to time and this divergence can continue in an infinite loop. To provide multi-variant recommendation lists, new items must satisfy the users.

Ant Colony Optimization is an effective algorithm for dealing with NP-hard problems Dorigo and Gamberdella (1997). Because of this vigorous aspect of the

algorithm, it has been applied in many recommender models. However, these models have investigated pheromone optimization on extensive graphs and tuning with complex reciprocal hyper-parameters. To illustrate, α (pheromone pressure) and β (heuristic influence) parameters should be tuned in some algorithms to understand their substantial effect of them. In most algorithms, the hyperparameter selections are chosen for all test users in the experiments, and the maximized parameters are determined by the average of the results. However, whether the systems are trained with the right hyper-parameters can be seen by waiting for the training result of the system. This causes a long evaluation time, especially for researchers operating huge datasets or models that require many hyperparameters.

3.1.2. Ant Colony Optimization

Ant Colony Optimization models are derived from the behavior of real ants to solve many optimization problems. Ants can discover the shortest path from a food source to the nest. While traveling, each ant deposits a chemical hormone, called pheromone on the ground and reflects the deposited pheromones by the other ants. Ant algorithm is a sample of algorithm belonging to swarm intelligence methods, based on collaboration between independent, distributed bots that can offer a new intelligent solution to the system. It is a suitable model for mimicking the behavior of users in recommendation systems. The process of obtaining new feature subsets from a few input features can be viewed as an optimization problem, and unclicked item predictions from clicked items agree with this definition in recommendation systems.

Algorithm 1: **Pseudocode for ACO**

```
Begin;
  Initialize pheromone amounts between nodes and set up parameters;
  Begin Loop
    Generate a solution for each ant;
    For each ant, estimate the fitness/cost;
    Determine the best ant for iteration;
    Update pheromone trails on nodes according to the best ant solution;
    If termination condition is true, exit from the loop;
  End Loop
End;
```

Figure 3.1. Pseudocode for ACO (Socha and Dorigo, 2008)

The ACO algorithm was presented in Algorithm 1 (Socha and Dorigo, 2008). The process with the ACO algorithm initializes nodes in a graph, obtains the weight values between the nodes at an unspecified time, and spreads ants randomly on these nodes. An ant *k* at *t* time, being in node *i* chooses the next node *j* with a probability given by the random proportional rule defined below Eq. (3.1)

$$probability_t^k(i,j) = \frac{\tau_t(i,j)^{\propto} * \eta(i,j)^{\beta}}{\sum_{k \in |u|} \tau_t(i,k)^{\propto} * \eta(i,k)^{\beta}} \quad (3.1)$$

where *u* is set of the feasible routes of *i*. After each ant walk, feasible routes are excluded by the last visited node. Once an ant has visited all nodes, it returns to its starting node. After evaluating all ant's tour costs in the current iteration, the pheromone values between nodes are updated as follows,

$$\tau_{t+1}(i,j) = (1-\rho) * \tau_t(i,j) + \sum_{i=1}^{m} \Delta\tau_k(i,j) \quad (3.2)$$

where ρ is the evaporation parameter between [0,1] and $\Delta\tau_k(i,j)$ is defined in Eq. (3.3)

$$\Delta\tau_k(i,j) = \begin{cases} Cost(k) * Q, & if\ edge\ (i,j)\ is\ a\ visited\ node\ for\ best\ ant, \\ 0, & otherwise, \end{cases} \quad (3.3)$$

Cost(k) is the amount of pheromone deposited on the edges of the solution visited by ant *k* and this cost is equal to the reciprocal of the cost of the solution constructed by ant *k*. *Q* is a constant to regularize the pheromones for the best solution. Therefore, a better solution is achieved by the higher amount of pheromone deposited by an ant.

3.1.3. Ant Colony Optimization in the Continuous Domain

Combinatorial optimization such as classic ACO deals with finding optimal combinations of available problem components and they attempt to find their optimal combination or permutation like in the (Travelling Salesman Problem) TSP problem. But some problems may be tackled with a combinatorial optimization that is not always convenient, especially if the bounds are wide, and the sensitivity of the parameters is high. In such cases, algorithms that optimize on continuous variables yield better results. Blum (2005) attempted to extend ACO algorithms for tackling discrete and continuous optimization problems.

There are two options for integrating ACO for continuous optimization problems. The first way uses a familiar approach to ant behavior and the second way carries the fundamental ACO graph structure to investigate it in the continuous domain. This evolution could be flawless by proper discretization or probabilistic sampling of a search space (Riadi, 2014).

Socha and Dorigo (2008), who introduced the continuous field ant colony optimization algorithm $ACO_{\mathbb{R}}$, used a Gaussian kernel probability density function (pdf) expression for the distribution model and presented the $ACO_{\mathbb{R}}$ as a meta-heuristic framework. In $ACO_{\mathbb{R}}$, given a problem with *n* decision variables, a vector $x_{ki} = \{x_{i,1}, x_{i,2}, x_{i,3}, \ldots, x_{i,n}\}$ represents probabilities from a probabilistic density function as a solution by an ant *k*, and f(*xi*) represents the objective function to

minimize (or maximize). In $ACO_{\mathbb{R}}$, each ant represents a row of the *Solution Archive*. During the iterations, the candidate parameter in the *Solution Archive* is ordered according to its objective function values. Each solution has an associated weight ω_k, which keeps the proportion of its solution quality on the whole. The weight of the *j*th solution is defined as:

$$\omega_j = \frac{1}{q\sigma\sqrt{2\pi}} e^{-\frac{(G(j)-\mu)^2}{2q^2\sigma^2}} \tag{3.4}$$

where *G(j)* is the value of the Gaussian function with argument *j*, μ is the distribution mean, σ is the standard deviation and *q* is the parameter for the deviation distance of the algorithm. When *q* is a small value, the high fit solutions are promoted, and with the increase of *q*, the probability becomes intensified. To implement the *pheromone motto* from the original ACO, after each iteration, the algorithm defines new μ and σ values to shift the probability distribution. Once the initial *Solution Archive* is constructed, iteration processes follow: Each ant selects a distribution from the solution archive with the asset of a fitness proportionate selection function such as the Roulette-Wheel algorithm, and the solution probabilities of each row are obtained by dividing all sums by themselves,

$$p(j) = \frac{\omega_j}{\sum_{r=1}^{k} \omega_r} \tag{3.5}$$

where $p(j)$ is the probability of the *j*th row in the *Solution Archive* set. In the iterations, after each ant creates a distribution similar to the *Solution Archive*, its quality is calculated based on the objective function and merged with the *Solution Archive*. After a sorting, the first *k* best solutions are selected and the others are discarded for forthcoming iterations. For example, for a maximization problem, the

Solution Archive constructed by *k* ants is ordered as descending. Hence $f(x_1) \geq f(x_2) \geq \cdots \geq f(x_k)$ and $\omega_1 \geq \omega_2 \geq \cdots \geq \omega_k$. The sample *Solution Archive* structure is given in Fig. 3.2

$$\begin{bmatrix} x_{11} & x_{12} & \dots & x_{1n} & f(x_1) & w_1 \\ x_{21} & x_{22} & \cdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ x_{k1} & x_{k2} & \dots & x_{kn} & f(x_k) & w_k \end{bmatrix}$$

Figure 3.2 The archive of solutions kept by ants

In the search process, the purpose of iterations is to find the best solution and converge the model.

After each iteration, the pheromone update strategy (like ACO) is performed by adding *k* newly generated solutions to the Solution Archive. After sorting the solutions, the worst *k* solutions were eliminated so that the total number of solutions in the archive remains *k* solutions. This method maintains the better solutions in the Solution Archive as a result of the effective guidance of ants in the search process for better quality.

In this study, we studied the RSs problems that we defined in Section 1.2 and founded on the $ACO_{\mathbb{R}}$ method, and we made new additions to this method to challenge the RSs problems.

3.1.4. AcoRec

In our model, we handled the problems related to the recommendations for cold-start users, personalized recommendations, over-specialization problems, and facilitating time complexity in the recommender systems. We introduced the AcoRec framework, which we produced using the ACO method to improve the variety of the user-item relations and to diversify the results for the users in the system. Based on ACO principles, AcoRec uses any item similarity/proximity model

as input and produces user-specific, probabilistic, and high-diverse recommendations for users based on their past purchases. As a meta-heuristic and hybrid framework, AcoRec explores diverse recommendations by which it could overcome the problems with pertinent recommendations for the cold-start users.

In our proposed model,

- AcoRec takes an item-based similarity model as input. The item-based similarity model could be pre-calculated in the background in an offline system.
- From the input model, AcoRec extracts the clicked (or preferred) item rows of the specific user, then combines rows and transforms them into an Lp norm vector for a low-dimension ant search. AcoRec verifies these norm spaces as the initial pheromone values (τ) and uses the input model as the heuristic for the items (η). The initial pheromone vector defines the user's current importance values on the items. Therefore, the model converges quickly and reduces the iteration counts, daemon actions, and the probability of stagnation.
- AcoRec optimizes the importance values of items for the specific user by using auto hyper-parameter tuning in the Continuous Ant Colony Optimization domain.
- After the model converged, AcoRec sorts out the most valuable items for the relevant user based on the expected choice of those items. Later, our model provided a top-N recommendation list of items that ranks the users' estimated predictions. These predictions could variate and differ from the previous estimates and this is the core phenomenon of our novel algorithm.

The evaluation of our novel model was carried out by comparing with MovieLens, Pinterest, and Yahoo datasets on different scenarios. We applied the near approaches which are popular algorithms in literature for the benchmark. Since our model is a heuristic approach, the recommended items are changed during the sessions, but we attempted to maintain the relevance and satisfaction of these items with the user heightened. The model we suggested needs a one-spatial vector for pheromones instead of two-dimensional pheromone graphs so that there become a decrease in the number of ants walked. AcoRec is also an algorithm suitable for parallelization with row-based user recommendations and ants running on multiple processors.

3.1.5. Stochastic Approach of AcoRec

AcoRec's constructs as a vector pheromone model can be easily adapted to a session or ongoing system for a user and try to predict users' interest in items using a Bayesian approach based on their previous clicks and adjust the user-based hyper-parameter to maximize the expectations.

In AcoRec, the probabilistic transition rule for the users, selected by ant ***k*** who mimics user ***u*** at ***t*** time is given in Eq. (3.6),

$$probability_t^k(u) = \tau(u)_t{}^{\propto} * \eta^{\beta} \quad (3.6)$$

where, $\tau(u)_t$ is equal to the pheromone values at *t* time on items for user *u*, η is the heuristic between the items by selected input model, α is the pheromone regularization parameter and β is the regularization parameter to adjust the heuristic model. These parameters determine the priori information of the items for users, and it is similar to the prediction of item-based models in general. Item-based models predict user scores for items in a primary way like in Eq. (3.7),

$$\text{predictions(u)} = \text{r}u * \text{S} \tag{3.7}$$

Let **S** be an m *x* m item similarity matrix and **r*u*** is a set of ***m*** items on which the user clicked. It is shown as **r*u*** = [**r*u*$_1$, ... , **r*u*$_\text{m}$]. If we accept the clicked items that users taste before by the means of pheromone traced items for users, AcoRec denotes **r*u*** vector as pheromone vectors and S as heuristic information between the by-items for further operations. We can also assume that pheromones could be carried by ants from (mimicking users) items to items.

In this situation, to construct posteriori pheromone vectors, we pick the rows of the items that the user clicked before from the item-item similarity model (column values will also be the same in symmetric matrices if it is a Hermitian matrix) and compose them into a low-rank vector to form the $L_\mathbf{P}$-*norm* from the columns of this subset matrix. The norms of the user clicked items means the user's actions as a pheromone vector (prior probabilities), and it is similar to the behavior of social networks. This is an initial pheromone interpolation, but we do offer a development that does not contradict the intuitive principles of the ACO algorithm. Priori pheromone values based on the $L_\mathbf{P}$-*norm* of clicked items gave good results, especially for recall values (see Section 4.1). Let **x*u*** = [**x*u*$_1$, ... , **x*u*$_\text{q}$] is a subset vector of **r*u*** which contains all clicked items belonging to user ***u,*** and ***q*** is the clicked item count. The formula for the $L_\mathbf{P}$-*norm* of these clicked items is shown below:

$$Lp(\text{r}u) = \left|\left|Su^{q*m}\right|\right|_p = \sum_{i=1}^{m}\sum_{j=1}^{q} S(j,i)^p \tag{3.8}$$

where **Su** is a subset matrix of **S** that only keeps the **x*u*** element rows which are clicked items of user ***u***, **S** item-similarity model, ***i*** is the column id in the item similarity model, and *m* is the total item size. In Eq. (3.8), when the ***p*** value is 1 this

means L_1-*norm*, and if the $\boldsymbol{p}$ value is equal to 2 it is equal to L_2-*norm* also known as *Euclidean Space*. To amortize the $L_{\mathbf{P}}$-*norm* vectors with the original clicks to infer how the similarity model responds to the user knowledge, we used Binary-Cross-Entropy (BCE). BCE is generally a utilizer as a loss function in classification tasks, but we applied it as a regularization for each value in the norm vector. The regularization of user clicked items are proven on-time estimate formula is given in Eq. (3.9),

$$\tau(u)_t = ru * \log(Lp(\mathrm{r}u)) + ((1 - ru)\log(1 - Lp\,(\mathrm{r}u)) \tag{3.9}$$

where $\tau(u)_t$ is the pheromone value of the items for user $\boldsymbol{u}$ at *t=0* time, ru is a binary vector with 1s if the user clicked item *i* and 0 otherwise. From Eq. (9) we discarded the unclicked items, to fill pheromone with only clicked items and optimized with below Eq. (3.10),

$$ru_i = \begin{cases} 1, & if\ user\ clicked\ item\ i, \\ 0, & otherwise \end{cases} \tag{3.10}$$

where ru_i is user click information about an item $\boldsymbol{i}$ for user $\boldsymbol{u}$, if a user clicked item *i* its value is 1, otherwise 0. Clicked item values amortized with L_p-*norm* values and user pheromone vector initialized with clicked item information only. A constant parameter initialization for the pheromones might be a marginally worse start for the optimal solutions because stagnation could act on more distant solutions. Pheromone initialization is useful for preventing stagnation, which is one of the main challenges in ACO algorithms (Dorigo & Gamberdella, 1997).

In the Ant Colony Optimization, the extremely significant influence provoking randomness in the search space is the pheromone model (Dorigo & Gambardella, 1997). Initial pheromone values for user clicked items are estimated

by Eq. (3.6), Eq. (3.8), and Eq. (3.9). In Eq. (3.6) the α parameter adjusting the tendency of pheromone values in ACO models and the β parameter choice, which controls the heuristic knowledge of the model are conditions that affect the profit of the model (Stützle et.al., 2011). After initializing the AcoRec pheromone values with the previous clicks of the user, we fixed the α parameter with α=1, which determines the pheromone bias in our model, and worked on the adjustment of the β parameter. We figured out that regulating heuristic knowledge with β and reducing the effect of bias increase the pheromone effect, or vice versa, decreasing the pheromone effect and increasing the tendency of heuristic knowledge. Parametric scale on heuristic knowledge, the overthrow of Euclidean norm data to popularity has been used in many algorithms, and successful effects have been seen (Nikolakopoulos et.al., 2017; Frolov & Oseledets et.al., 2019; Paudel et.al., 2016). The scaling on the heuristic **S** matrix defined with Eq. (3.11)

$$\boldsymbol{S} = \begin{bmatrix} s_{11} & s_{12} & \cdots & s_{1m} \\ s_{21} & s_{22} & \cdots & \vdots \\ \vdots & \vdots & \ddots & \vdots \\ s_{m1} & s_{m2} & \cdots & s_{mm} \end{bmatrix} * \begin{bmatrix} \|s_1\|_F & 0 & 0 & 0 \\ 0 & \|s_2\|_F & 0 & 0 \\ 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & \|s_m\|_F \end{bmatrix}^{\beta} \tag{3.11}$$

where β is the scaling parameter and $\{\|s_1\|_F, \ldots, \|s_m\|_F\}$ are Frobenius norm (L_2-norm) of each column in the **S**. The scaling parameter is used to reduce and increase the effect of high norm values in popular items. If we re-insert the pheromone values, we obtained the scaled Heuristic model by Eq.(3.9) and Eq.(3.11) in Eq. (3.6) and we got the following formula Eq. (3.12),

$$probability_t^k(u) = \tau(u)_{1m}{}^{\propto} * S_{mm} * \text{Diag}(\{\|s_1\|_F, \ldots, \|s_m\|_F\})^{\beta} \tag{3.12}$$

In our experiments, we observed that (see Section 4.1), the maximized β parameter search would reflect the user's personality. When the β parameter has a

negative value, rare items can also be highlighted for the user. This parameter could vary for individual users on the system according to their taste.

The parameter that had the best fitness value could become a scale factor for the users in our study. However, discrete probabilities may not hand over certainty while searching a hyper-parameter. That's why the optimization dilemmas of continuous fields have redirected into a new direction for ant colony optimization research. In the continuous domain, instead of running a discrete probability distribution, a pdf is employed to sample the probabilistic hyper-parameters. One can think of a node in a conventional ACO problem as a local parameter in the Gaussian Distribution. We discussed finding the maximized value of the β parameter in the $ACO_{\mathbb{R}}$ domain and explained our version of $ACO_{\mathbb{R}}$ about finding the ideal β parameter in **Algorithm 2**. Sensitively neighboring points in a continuous domain offer close results, and we can investigate the maximized β parameter on a continuous field in a stochastic way. In our model, each ant samples a pdf in the Gaussian Distribution, and these points are seen as candidate β parameters. By using the sampling from $G(x) = N(\mu,\sigma)$ in the beginning, we initialize $\mu=0$, $\sigma=1$, and each ant samples a random point as β. Then each ant's probability values are estimated for the current iteration with Eq. (3.12). AcoRec uses a non-linear normalization function for each ant's probability vector and ordered descending and trimmed first N item scores in each vector. If the validation items are in these trimmed probability vectors, we binarized activation scores of them by keeping their position and discarding others as zero. As an evaluation for the ant solutions, AcoRec uses R-Score@100 for evaluating the best β for the validation items in Eq. (3.14), which we explained in Section 3.4 and we cloned probability vectors by multiplying the position scores of R-Score@100. Now we have two multi-class matrices, denoting probability vectors and evaluation metrics' hit values which were synchronized with probability. To estimate the likelihood of the ant solutions, we used a similar approach in Eq. (3.9) as a fitness function. We used matched values from the metric

as labels **y*k*** and ant solutions for predicted probability values **p(y*k*)** for ant ***k*** in Eq. (3.15) for every ant solution.

$$\boldsymbol{max}(u) = \max_{1 \le i \le N} probability_t^k(u) \tag{3.13}$$

$$\mathbf{y}\boldsymbol{k}(\boldsymbol{i}) = \begin{cases} \dfrac{1}{2^{\frac{j-1}{\alpha-1}}}, if\ item\ i\ at\ the\ pos\ j\ is\ in\ validation\ list \\ 0, \qquad\qquad otherwise \end{cases} \tag{3.14}$$

$$\mathbf{p}(\mathbf{y}\boldsymbol{k}) = \max_{1 \le i \le N} normalization(probability_t^k(u)) \tag{3.15}$$

$$Fitness_k = \sum_{i=1}^{N} \mathbf{y}\boldsymbol{k}(\boldsymbol{i}) * \log(\mathbf{p}(\mathbf{y}\boldsymbol{k}(\boldsymbol{i}))) + ((1 - \mathbf{y}\boldsymbol{k}(\boldsymbol{i}))\log(1 - \mathbf{p}(\mathbf{y}\boldsymbol{k}(\boldsymbol{i})))) \tag{3.16}$$

With the fitness function in Eq. (3.16), we sort the ants in descending order according to their solution quality. We used this process to initialize the solution matrix in $ACO_{\mathbb{R}}$. Solution space rows are equal to *ant_size* in our approach and each row has a sampled β value cell and a cell that keeps the fitness value. After constructing the solution iteration process, we trained the best hyper-parameter search on the user evaluations. At each iteration, according to the current archive of the solutions, μ and σ are to be estimated by the current population and with their weights. This process shifts the distribution of the best quality μ and β at the same time. The core of ACO algorithms depends on pheromone evaporation. These phenomena are implemented in our algorithm as the shifting and squeezing distribution. In each iteration, the deviation of the distribution is tightened. Each ant discovers β from the new $N(\mu,\sigma)$ Gaussian distribution, and these β values are sorted

by their qualities and we construct an iteration solution archive and merge it with the main solution archive. Then all solutions are sorted and the best top-*N* solutions according to their weight are chosen for the next iterations. At the end of iterations, the best quality β or the mean of solution archive is chosen for the Eq. (3.16) user predictions.

Algorithm 2: **AcoRec**

Inputs: Item Similarity Model $\mathbf{S} \in \mathbb{R}^{m*m}$, Click vector of user $\mathbf{ru} \in \mathbb{R}^{1*m}$
Scale ← Frobenius norm of columns of **S**, Diag({$||s_1||_F$,...,$||s_m||_F$)
μ ← 0
σ ← 1
tol ← 1e-4
as ← ant size and archive size
it ← iteration count

Output: $Predictions_u$ ← Predictions of user ***u*** for items

```
Compute Lp(ru) from Eq.(8)
Construct SolutionArchive(1...as) ← {}
for i ← 1,2, …,it do
  for k ← 1,2, …,as do
    // random β variable using the Gaussian distribution with mean μ and deviation σ
    rβ = N(1|μ,σ)
     // pheromones for each ant, which is given in Eq.(12)
    Pheromones_k = Lp(ru) * S * Scale^rβ
    // Fitness_k value for each ant using Eq. (20)
    if Fitness_k > SolutionArchive(as) then
      SolutionArchive(as) = Fitness_k
      sort(SolutionArchive)
    endif
  endfor
  //Pheromone update strategy
  μ ← μ(SolutionArchive)
  σ ← σ(SolutionArchive)
  if σ < tol then
    exit
  endif
endfor
// Best Solution
β = μ(SolutionArchive(1))
return Lp(ru) * S * Scale^β
```

Figure 3.3. Pseudocode for AcoRec

3.1.6. Heuristic Base of AcoRec and Item Based Model Selection

The distance between nodes is crucial for ants to choose their later positions. In the TSP problem, it is beneficial to have a short distance. In the ACO-based recommender systems, the distance between nodes is represented by the similarity/proximity between items (or users). For this similarity, distance measurements in inter-nodular Euclidean space are preferred. We designed our model as low-dimensional and determined the user's interest in items as heuristic data rather than considering the distance between nodes. The relation between items is controlled in many respects. These relationships could be in various forms such as similarity, proximity, dissimilarity, or correlation, and can be shown by specific methods. CF Based Similarity Models acknowledge the collaborative benefit of the items. CBF Similarity models zoom in on related items dealing with the metadata (demography, mood, etc.) of the items. Graph Similarity models are based on the relations in the user-item network structure. Time-Based models track the time sequences of the purchase for the items. Latent-Factor Based Models extract hidden components from low-rank computations. Demographic Models care about collaborative behaviors in the same geographic locales.

In this study, we evaluated three well-known item-based similarity measures for computational simplicity and popularity; Let $\mathbf{S^{m*m}}$ be the similarity matrix, ***i*** and ***j*** be the two items, $\mathbf{v_i}$ and $\mathbf{v_j}$ be the column vectors of these items.

Dot Product (Dotp); Dot-product similarity of two items is equal to the inner product of these item vectors.

$$\mathbf{S}_{\text{Dotp}} = \left|v_i \cap v_j\right| = \overrightarrow{v_i} * \overrightarrow{v_j} \tag{3.17}$$

Cosine (COS); The Cosine similarity of the two items is the angle between their rating vectors. It is estimated by the inner product of these item vectors by dividing by vector norms multiplication.

$$\mathbf{S}_{\text{Cos}} = \frac{|v_i \cap v_j|}{\sqrt{|v_i| * |v_j|}} = \frac{\overrightarrow{v_i} * \overrightarrow{v_j}}{||\overrightarrow{v_i}|| * ||\overrightarrow{v_j}||} \tag{3.18}$$

Jaccard (JAC); The Jaccard similarity between two items is defined as the ratio of the number of users that co-rated items based on the number of users who rated at least either *i* and *j* items.

$$\mathbf{S}_{\text{Jac}} = \frac{|v_i \cap v_j|}{|v_i \cup v_j|} = \frac{\overrightarrow{v_i} * \overrightarrow{v_j}}{||\overrightarrow{v_i}|| + ||\overrightarrow{v_j}|| - \overrightarrow{v_i} * \overrightarrow{v_j}} \tag{3.19}$$

3.2. A Deterministic Perspective, z-scoREC and ImposeSVD

To overcome RSs challenges, the CBF or CF methods have been applied to the recommendations in sparse datasets in various studies (Adomavicius & Tuzhilin, 2005). The application of CF method has shown successful results with Singular Value Decomposition (Golub & Van Loan, 2013), which is one of the low latent factor approximation techniques based on matrix factorization.

The effectiveness of SVD-based models results from their ability to uncover latent factors between users and items, which are hard in traditional nearest-neighbor approaches (Sarwar et.al., 2000). But SVD-based models have several limitations including high costs for estimations and difficulty with the algebraic calculations because of the incomplete matrices. As a result of developing hardware technology and optimized linear algebra libraries, the use of algebraic SVD-based models has become popular in RS as a result of their successful results in providing recommendations.

One of these SVD-based models incorporated in the RS technologies is the PureSVD model developed by Cremonesi et al. (2010). The PureSVD model offers low-rank factors estimated with SVD after empty cells in the user-item matrix are

imputed as ‘zero’ at the outset. One of the advantages of the PureSVD is its easy implementation with linear algebra libraries written in many programming languages.

Many researchers have used the PureSVD as a benchmark algorithm (Cremonesi et.al., 2010; Kabbur et.al., 2013; Cheng et.al., 2014; Wu et.al., 2016; Kang et.al., 2016; Nikolakopoulos et.al., 2017; Christakopoulou et.al., 2018; Nikolakopoulos et.al., 2019; Frolov & Oseledets, 2019) and agreed on the fact that the PureSVD with its basic structure is a successful non-parametric model to be applied to RSs.

However, the inefficiency of the PureSVD in cold-start situations has led researchers to investigate different solutions and improvements in the model (Nikolakopoulos et.al., 2017; Nikolakopoulos et.al., 2019; Frolov & Oseledets, 2019).

With regards to the cold-start problem, EigenREC developed by Nikolakopoulos et al. (2017) produced faster and more accurate results in high-dimensional data by replacing the SVD-based model of the PureSVD with the *Eigenvalue Decomposition*.

The HybridSVD model (Frolov & Oseledets, 2019) has successfully exploited the PureSVD's disadvantages stemming from CF by embedding side-information to provide better solutions to cold-start and sparsity problems.

Another research (Christakopoulou et.al., 2018) also shows that the PureSVD is also suitable for parallel operations in high-dimensional data with the *Golub-Kahan-Lanczos Bidiagonalization* method (Golub & Van Loan, 2013).

The main focus of all these studies is to improve the effectiveness of the PureSVD model in different problems encountered in RSs without disrupting its simple structure. However, as Cremonesi et al. (2010) stated ‘there are still several unexpected ways that may improve PureSVD'. One of the ways they suggest is to ‘optimize the value imputed at the missing entries' instead of assigning zero as a value.

Ghazanfar and Prugel (2013) also suggested that a particular amount of imputation taken with care could enhance the quality of recommendations.

Adding to the line of the studies improving the PureSVD model, this study aims to suggest novel models for the sparse datasets that will improve the recommendations for cold-start users and provide long-tail items in the recommendation lists by finding out good-working imputing strategies.

In these models**;**

- We propose a novel basic asymmetric item weight matrix based on Gram-matrix. Unlike conventional similarity methods, we undermined symmetry to catch various relations between the elements. Gram-matrix was shifted with L_1-*norm* of items that were propagated from the item matrix. The shifting process penalized poor ratings, changed the negative entries to zero, and broke the symmetry in the Gram matrix. The parameter for shifting could adjust the different relations without disrupting the identity of mass data. Later, we employed the Z-Score normalization, ignored negative values, and found a disagreement between the non-symmetric weight matrix and the regular Gram-matrix. The shifting and normalization were element-wise; and thus, were not heaped and time-consuming. With the cooperation of this fresh weight matrix, we designed a baseline prediction matrix. We called this model z-scoREC. The imposed matrix, reproduced from z-scoREC, maintained new relationships between the users and the items but yet sparse, which makes it computable for the big data.
- We used z-scoREC predictions toward the imposed matrix. But before decomposing this imposed matrix we came up with some normalization and regularization processes on these priori predictions. After decomposing low-rank latent factors with SVD libraries from this imposed matrix, we estimated a new enriched prediction matrix and

acquired valuable top-N recommendation lists for users. We called this method ImposeSVD.

- When we evaluated the ImposeSVD and the z-scoREC on common popular datasets for the cold-start scenarios and long-tail item recommendations using the well-known evaluation metrics, we found that our models outperformed similar state-of-art methods. Additionally, z-scoREC surprisingly gave closer results to ImposeSVD by using it in the basic item-based prediction model.

3.2.1. Notations for z-scoREC and ImposeSVD

In the rest of this section, vectors and matrices are denoted in bold letters. We used bold capital letters for matrices and bold lowercase letters for vectors. In addition, ***u, v*** represent users and ***i, j*** represent items. We denoted the user-item rating matrix as **R**. The dimension of **R** is *n x m*, where ***n*** is the number of users and ***m*** is the number of items. The rating is given by user ***u*** to item ***i*** is denoted with $\mathbf{r}_{ui}$ in **R**. $\tilde{\mathbf{R}}$ represents the rating prediction matrix and $\tilde{\mathbf{r}}_{ui}$ denotes the predicted rating score of the user ***u*** for item ***i***. $\acute{\mathbf{R}}$ represents the impose matrix used for embedding to the **R**. $\check{\mathbf{R}}$ represents the decomposition matrix of **R** or $\acute{\mathbf{R}}$. $\grave{\mathbf{R}}$ denotes the final prediction matrix that is the union of the original **R** and $\acute{\mathbf{R}}$ matrix, $\grave{\mathbf{R}}_u$ represents the row vector of user ***u*** in $\grave{\mathbf{R}}$.

Item-based CF models aim to predict users' ratings for a specific item by the dot product of the user selections with item-item weights. For the estimation of the $\tilde{\mathbf{R}}$, the common formula of the prediction matrix is given in Eq. (3.20)

$$\tilde{\mathbf{R}} = \mathbf{RK} \tag{3.20}$$

where $\mathbf{R} \in \boldsymbol{R}^{n*m}$ could be a binary purchase (or a ranged scalar rating matrix, or a listen to count) matrix, $\mathbf{K} \in \boldsymbol{R}^{m*m}$ is an item-item weight matrix that defines the

proximity of the items between themselves. Establishing an item-item weight matrix is essential for the model's progress and this model must handle the typical challenges in RSs. It is also important for this matrix to be satisfying and computable with parallel measurements.

K could be a similarity matrix determined by correlation methods such as *Dot-Product, Cosine-based, Jaccard,* or *Pearson-correlation*. These correlation methods are adopted in many CF, CBF, or Hybrid models (Sarwar et.al., 2001; Deshpande & Karypis, 2004; Frolov & Oseledets, 2019). For instance, the *itemKNN* method was used by Deshpande and Karypis (2004) to find out the most similar *k* items for each row in the item-item similarity matrix estimated by Cosine-Similarity or Conditional Probability-Based Similarity. The researchers dismissed all items in every column except for the *k* items and constructed a prediction matrix for top-N recommendations for each user in the **R** matrix. The *itemKNN* method; however, suffers in sparse datasets and cold-start situations because it only considers the existing collaborations. Also, *k* represents the number of neighbors or the threshold value and its optimal could differ between the items in the same dataset. As a result, the same *k* value along rows and columns can be overfitting or underfitting for different users or items (Ning et.al., 2015). Another popular approach, SLIM trained **K** as a coefficient matrix and made predictions with the help of linear regression models such as *ElasticNet* and *Lasso* (Ning and Karypis, 2011). Kabbur et al. (2013) created two item factors with the model they called FISM with help of loss functions by sampling the previous clicks users and training this obtained data. Fouss used graph relations based on kernel similarity methods to define suitable item-item weight matrices (Fouss et.al., 2006). In addition to these, the PureSVD and its generalized methods constructed item-item weights from latent factors of items based on the use of low-rank item factors estimated by the decomposition algorithms (Cremonesi et.al., 2010; Nikolakopoulos et.al., 2017; Nikolakopoulos et.al., 2019; Frolov & Oseledets, 2019).

3.2.2. z-scoREC Model Definition

In our model, we proposed:

- A sparse but qualified **K** weight matrix to generate efficient recommendations with a low cost for huge data.
- An asymmetric **K** weight matrix to capture individual relations between items without needing explicit data.
- To trivialize meaningless values by naturally, on the contradictory of neighborhood approaches.
- A less parametric model which is trainable and adaptable easily.

The dot product of users and items vectors is the simplest way to obtain an item-item similarity matrix. It is also called co-citation, Gram-matrix and in the simple form, it is estimated as in Eq. (3.21)

$$\mathbf{K} = \mathbf{R}^{\mathbf{T}}\mathbf{R} \tag{3.21}$$

where $R \in \mathbb{R}^{n*m}$ and R>=0, then K is a symmetric, positive, and semi-definite matrix. In a real scenario, R is mostly sparse and includes null values, which are insufficient for the Gram-matrix.

Assume that our $\mathbf{R} \in \boldsymbol{R}^{n*m}$ matrix has binary values if a user clicked a specific item and zero values if not clicked. Then, we compute the item-item similarity matrix as in Eq. (3.22), by multiplying $\mathbf{R}^{\mathbf{T}}$ with the element-wise shifted version of **R** matrix where λ is a real value in the range [-1,1] and **e** is an m*1 unit column vector having 1 in all elements to perform element-wise shifting operation.

$$\mathbf{K} = \mathbf{R}^{\mathbf{T}}(\mathbf{R} - \lambda\mathbf{e}) \tag{3.22}$$

This shifting ($\mathbf{R} - \lambda\mathbf{e}$) subtracts λ from all elements in **R** and transforms values of **R** so that matrix elements become real-valued, and non-clicked item ratings become negative.

For a given **R**, if we transpose $\mathbf{R^T R}$, then $(\mathbf{R}^T\mathbf{R})^T = \mathbf{R^T}(\mathbf{R^T})^T = \mathbf{R^T R}$, thus we know that Gram-matrix $\mathbf{R^T R}$ is symmetric with a primitive rule of equality of its transpose. However, in Eq. (3.22) we multiply $\mathbf{R^T}$ with the shifted version of **R**, and if we apply distribution rule to Eq. (3.22) we obtain Eq. (3.23), and we can say that the **K** matrix cannot guarantee to be symmetric and could be asymmetric if the matrix dimensions are not equal ($m \neq n$). This asymmetry provides the use of new relations in the weight matrix. When we shift the **R** matrix with **λ**, $\mathbf{R} - \lambda\mathbf{e}$ grew into a dense matrix that leaves null values, and computation of Eq. (3.22) became extremely costly. Therefore, we use Eq. (3.23) to estimate **K**.

$$\mathbf{K} = \mathbf{R^T R} - \mathbf{R^T}\lambda\mathbf{e} \tag{3.23}$$

We obtained an improved formula and separate the shift operation from Gram-matrix estimation. The diagonal $\mathbf{d} = \text{Diag}(\mathbf{R^T e})$ yields the column L_1-*norms* of **R** (Fouss et.al., 2016) and we get Eq. (3.24).

$$\mathbf{K} = \mathbf{R^T R} - \lambda * \mathbf{d} \tag{3.24}$$

where vector **d** is multiplied by λ and subtracted from each column of the Gram-matrix.

LEMMA 1. If $\mathbf{R} \in \boldsymbol{R}^{n*m}$ is a **binary** interaction matrix consisting of 1s and zeros, $\mathbf{R^T}$ is the transpose of **R** and $\mathbf{e} \in \boldsymbol{R}^{m*1}$ is full of 1s vector with the column size of $\mathbf{R^T}$, and not for only square matrices also includes non-square matrices then,

$\text{Diag}(\mathbf{R^T e}) = \text{Diag}(\mathbf{R}^T\mathbf{R})$

PROOF. Proof of Lemma 1 is presented in Appendix A.

By using Lemma 1, if $\mathbf{d} = \text{Diag}(\mathbf{R}^T\mathbf{e})$, then $\mathbf{d} = \text{Diag}(\mathbf{R}^T\mathbf{R})$ and we obtain Eq. (3.25).

$$\mathbf{K} = \mathbf{R}^T\mathbf{R} - \lambda * \text{Diag}(\mathbf{R}^T\mathbf{R}) \tag{3.25}$$

Now if we denote $\mathbf{R}^T\mathbf{R}$ as $\mathbf{G}$ Eq. (3.25) can be written as in Eq. (3.26)

$$\mathbf{W} = \mathbf{G} - \lambda * \text{Diag}(\mathbf{G}) \tag{3.26}$$

In a structure view, the shifting operation on the second matrix (which is the same as the matrix for the Gram-matrix estimation) is performed as row-based degree shifting on Gram-matrix. The shrinkage ratio is obtained by λ and if $\lambda=0$, our **W** matrix is pure Gram-matrix. It is also established that the regularization parameter λ between [-1,1] behaves as a penalizing term on neighborhood degrees. With $\lambda=1$ value, **W** is transformed into an unsigned *Laplace Operator Matrix*, which is a derivative of the *Laplacian Matrix* (Fouss et.al., 2016, p.18). We should notice a revised weighted Laplacian Matrix, as recognizing the Gram-matrix like an adjacency matrix of an undirected weighted graph. Laplacian operators, especially as a model used in image filtering, prevent noise from becoming dominant in the data. After adding noises to our data by shifting it, then we applied the whitening process. When we hold our model as a transparent auto-encoder system with non-hidden layers, we obtained **W** by including noise to the Gram-matrix in the first layer, and we demand to remove the noise from the data in the second layer. For kernel normalization, many methods are used, such as vector normalizing, vector centering, Z-Score, or Tanh estimators. As pointed out in the EASE^R article, many CF models are vanilla auto-encoder implementations (Steck, 2019).

Z-Score is an extremely impressive normalization process when handled with the real data set. Z-Score precisely eliminates noise from the data and reduces

variance, which is preferable in regression analysis. The primary formula of Z-Score normalization is given in Eq. (3.27).

$$\mathbf{K} = (\mathbf{W} - \boldsymbol{\mu}) / \boldsymbol{\sigma} \tag{3.27}$$

where $\boldsymbol{\mu}$ is the column means of $\mathbf{W}$ and $\boldsymbol{\sigma}$ is the column-based standard deviations of $\mathbf{W}$. Final step of the estimation produces an asymmetric dense matrix with lots of negative values if $\lambda \neq 0$. One of the reasons we adopt Z-Score normalization is that, at the end of the normalization, the values are excluded by holding an optimal threshold position, as the values can be observed in the $[-\infty, +\infty]$ range. The dense coefficiency matrix has rich relations but performance issues with evaluation costs. Removing these negative values will keep sparsity that eliminates insignificant relations according to our model.

Z-Score performs the whitening process on the weight matrix in our model, on autonomous columns, in parallelizable and with a low-cost estimation. In deep-learning models, a similar technique to Z-Score, which is mentioned Batch-Normalization, enables faster training time of the network and provides stable results for layers. In addition, we can consider it as a linear regularization model similar to SLIM, since we apply operations on Gram-matrix and normalization matrices as column-based regression and vector operations independent of each other.

Contrary to the SLIM (Ning and Karypis, 2011) and EASE^{R} (Steck, 2019) models, we don't drop diagonals and we observed that the zero diagonal weight matrix does not affect the results for our model in the experiments. The prediction model is based on the generalized Eq. (3.20) formula. $\tilde{\mathbf{R}}$ is the prediction matrix for all users. Fig. 3.4 presents an example that shows how our proposed method calculates weight matrix $\mathbf{K}$, which is used for the estimation of the prediction matrix.

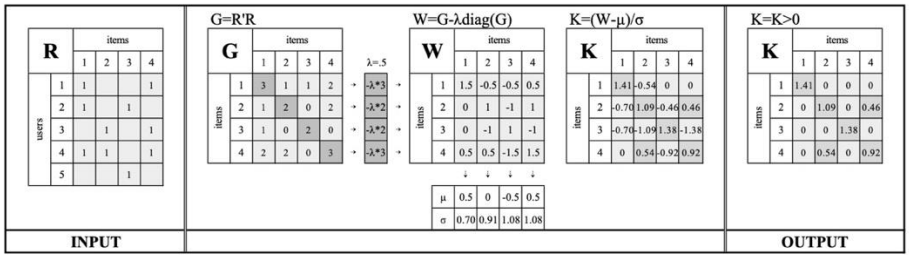

Figure 3.4 The architecture of the proposed item similarity matrix

As illustrated in the example in Fig. (3.4), **R** interaction matrix consists of 1s, and null values are considered as zero. We perform the $\mathbf{R}^{\mathbf{T}}\mathbf{R}$ to create the **G** item-item similarity matrix from the **R** matrix. It can be recognized that each diagonal element g_{ii} in the **G** matrix corresponds to the sum of the elements in the associated column i in the **R** matrix. That is, the L_1 norm of each column i of the **R** matrix is equal to each diagonal g_{ii} element of the **G** matrix. We obtain the **W** matrix by subtracting $\boldsymbol{\lambda} * g_{ii}$ from each element in row i of **G** matrix. As an example, let $\boldsymbol{\lambda}$ be equal to 0.5, then 0.5*3 which is equal to 1.5 is subtracted from the elements in the 1st row, 1 is subtracted from the elements in the second row, etc. The **W** matrix is Gram-matrix at λ=0 and Laplacian-matrix at λ=1. In the **W** matrix, $\boldsymbol{\mu}$ defines the column means and $\boldsymbol{\sigma}$ defines the column standard deviations. The **K** matrix is computed by applying z-score normalization to each column of **W** matrix. After the normalization step, all negative values are converted to 0 in the **K** matrix as in Fig. (3.4). We called this method **z-scoREC**, and Algorithm 3 given below summarizes the steps applied to compute the **K** matrix, which is then used to estimate the user ratings $\tilde{\mathbf{R}}$ for the items by employing Eq. (3.20).

Algorithm 3 : **z-scoREC**

Input: Ratings Matrix $\mathbf{R} \in \mathbb{R}^{n*m}$, $\boldsymbol{\lambda}$ shrinkage value

Output: $\mathbf{K} \in \mathbb{R}^{m*m}$ weight matrix

// Gram-matrix, $O(n*m^2)$

1: $\mathbf{G} \leftarrow \mathbf{R}^T\mathbf{R}$

// subtraction of matrix elements are elementwise, $O(m^2)$

2: $\mathbf{W} \leftarrow \mathbf{G} - \lambda$*Diag**(G)**

// column-means $O(m^2)$, column-std. deviations $O(m^2)$ and

// Z-Score computation for each element, $O(m^2)$

3: $\mathbf{K} \leftarrow$ Z-Score(**W**)

// controlling every item because of asymmetric matrix, $O(m^2)$

4: **for** $i,j \leftarrow 1,2, \ldots,m$ **do**

5: **if** $[\mathbf{K}]_{ij} \leq 0$ **then** $[\mathbf{K}]_{ij}=0$

6: **end**

7: **return K**

Figure 3.5 Pseudocode for z-scoREC

3.2.3. Computational Complexity of z-scoREC

As it can be seen from Algorithm 3, the overall time complexity is $O(n*m^2)$. In Line 1, we estimated $\mathbf{R}^T\mathbf{R}$ which is the matrix product of two $m*n$ and $n*m$ matrices that can be computed in $O(n*m^2)$ time, and we get the $\mathbf{K}^{m*m}$ weight matrix between items. Note that the transpose of **R** matrix can be computed in at most $O(n*m)$ time. In Line 2, element-wise multiplication $O(m^2)$ and subtraction $O(m^2)$ are applied for **K**, which corresponds to the $O(m^2)$. In Line 3, we applied *Z-Score* normalization to the **W** weight matrix. In these processes; if we analyze a column, first we find the sum of the column in $O(m)$ time and we divide it into number of elements in the column in $O(1)$ time. Computing the mean of each column is done in $O(m)$ time, and for the whole matrix, this process takes $O(m^2)$ time. In finding the standard deviation of a column we subtract each element of the column from the column mean value then square the difference and sum all differences with $O(m)$ time, dividing it by the element count of the column and getting the root square is in $O(1)$ time. Finding standard deviation is applied for all columns and this corresponds to the $O(m^2)$ time. For Z-Score normalization, we subtract every element from its

column mean value and then divide this result by its column standard deviation value. Z-Score normalization of all elements in the matrix takes $O(m^2)$ time. In line 5 we removed negative values. This element-wise operation is done in $O(m^2)$ time complexity. In line 7, we multiply our novel weight matrix **K** with the original rating matrix and we got predictions. This process is made in $O(n*m^2)$ complexity for n users and m items.

3.2.4. ImposeSVD: Imposing SVD for cold-start recommendations

ImposeSVD is our proposal to estimate user ratings for items in cold-start cases. In this method, we apply SVD to the estimated rating matrix $\tilde{\mathbf{R}}$ that is imputed by using the weight matrix **K** computed by our z-scoREC algorithm to make a better recommendation for cold-start cases. The details of our proposal are explained in the below subsections.

3.2.5. SVD Analysis

Let $\mathbf{R} \in \boldsymbol{R}^{n*m}$ be a user-item rating matrix. In SVD based recommender models, the main idea is to decompose the **R** user-item matrix into low-rank matrices $\mathbf{P}^{n*f}$ and $\mathbf{Q}^{f*m}$ where $f<m$ and $f<n$. Let **P** be a matrix of *'user-factors'*, and **Q** is a matrix of *'item-factors'*. The basic prediction matrix $\tilde{\mathbf{R}}$ is formed from user factors and item factors and it can be computed as shown in Eq. (3.28);

$$\tilde{\mathbf{R}} = \mathbf{P}\mathbf{Q}^{\mathbf{T}} \tag{3.28}$$

PureSVD imputes null values with zeros in **R** and decomposes the $\boldsymbol{f}$ dimensional latent factors of the **R** matrix with $\boldsymbol{n}$ users and $\boldsymbol{m}$ items. When we decompose **R** to the singular values and unitary matrices, with the help of advanced linear algebra libraries, we get **U**, **Σ**, and **V** matrices from Eq. (3.29),

$$\check{\mathbf{R}} \cong \mathbf{U}_{n*f}\, \mathbf{\Sigma}_{f*f} \mathbf{V}^{T}_{m*f} \tag{3.29}$$

The matrices **U** and **V** are orthonormal matrices and **Σ** contains *f* singular values. Consider that rows in the **R** matrix denote users, and columns are items, then in the decomposition of the **R** matrix, **U** is user factors because of the similar dimensions, and **V** is the item factors matrix. From Eq. (3.28) and Eq. (3.29), we have the following equations Eq. (3.30) and Eq. (3.31);

$$\mathbf{P= U\Sigma,\ Q=V} \tag{3.30}$$

$$\check{\mathbf{R}} = \mathbf{U\Sigma V}^{T} = \mathbf{PQ}^{T} \tag{3.31}$$

where **U** and **V** are orthonormal matrices, and then we can easily get Eq. (3.32)

$$\mathbf{P = U\Sigma = RQ} \tag{3.32}$$

Replacing **UΣ** in Eq. (3.29) with **RQ** in Eq. (3.32) gives Eq. (3.33).

$$\tilde{\mathbf{R}} = \mathbf{RQQ}^{T} \tag{3.33}$$

where $\tilde{\mathbf{R}}$ is a simple prediction matrix for all users. Predictions of user $\boldsymbol{u}$, which is $\tilde{\mathbf{r}}_{\boldsymbol{u}}$, could be estimated with the user's row vector in **R** with Eq. (3.34)

$$\tilde{\mathbf{r}}_{\boldsymbol{u}} = \mathbf{r}_{\mathbf{u}}\mathbf{QQ}^{T} \tag{3.34}$$

In Figure 3.6 we show that how the PureSVD makes predictions. PureSVD imputes null values as zeros and decomposes this completed matrix with low ranks. PureSVD only takes the item factors and there is no need for the **U** for the predictions. That is one of the most persuasive features of the PureSVD algorithm because, for new users, there is no need to re-decompose **R**.

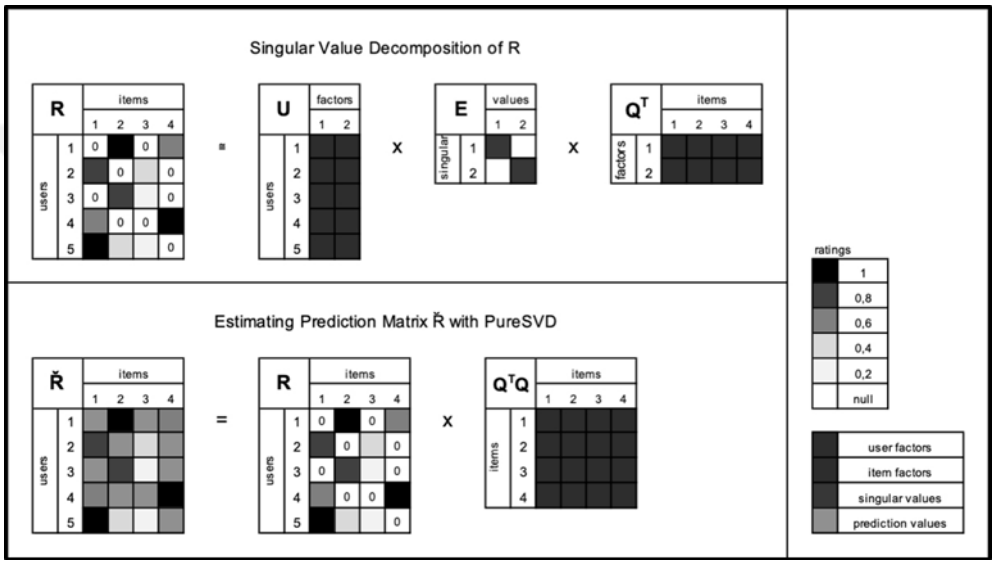

Figure 3.6 Rating prediction example with a simple matrix for PureSVD algorithm

However, analyses about PureSVD show that it's insufficient in cold-start and data unavailability situations. EigenREC showed that the efficiency and quality of PureSVD depend on the precision of the item factors from the decomposition (Nikolakopoulos et.al., 2017; Nikolakopoulos et.al., 2019). The authors concluded that "*the PureSVD algorithm's recommendation model is based on the induction of recommending similar items the user previously liked"*, and meanwhile the connections in the $\mathbf{QQ}^T$ matrix, which are the latent component factors of PureSVD, could be obtained from the *Eigen Decomposition* of the $\mathbf{R}^T\mathbf{R}$ matrix. Another approach, HybridSVD, has suggested that embedding side-information could recover PureSVD in cases where it is inadequate (Frolov & Oseledets, 2019).

3.2.6. Motivation behind ImposeSVD

We figured out how the PureSVD handles and produces recommendations when the sparsity decreases in the same dataset. Our evaluation scenario is to simulate the evolution of a recommender system with time. Based on this scenario, we used a common dataset; namely, MovieLens 1M (Harper & Konstan, 2015) for movie recommendations. This dataset has 6040 users, 3952 items, 1M ratings, and about 95% sparsity as shown in Table 1. We denoted this dataset as $\mathbf{Tr_1}$. Then, we randomly selected 66% of the ratings from $\mathbf{Tr_1}$ and named this subset $\mathbf{Tr_2}$. We continued to create subsets by selecting the randomly chosen 66% of the ratings from the parent subset until we got four subsets. At the end of the process, we got related coherent five training sets named $\mathbf{Tr_1} \supset \mathbf{Tr_2} \supset \mathbf{Tr_3} \supset \mathbf{Tr_4} \supset \mathbf{Tr_5}$ in which each set subsequently subsumes the following sets. Then, we created a test set consisting of full ratings for users (maximum rating) from the smallest training set ($\mathbf{Tr_5}$) and removed this test set from all training sets. We calculated the $\tilde{\mathbf{R}}_1$, $\tilde{\mathbf{R}}_2$, $\tilde{\mathbf{R}}_3$, $\tilde{\mathbf{R}}_4$, and $\tilde{\mathbf{R}}_5$ prediction matrices from their own training sets by using Eq. (3.33) and adopted the evaluation method in (Cremonesi et.al. 2010) to evaluate the quality of the top-N recommendations (shown as @N in Fig. 3.7) in the prediction matrices. To estimate the quality of the lists at different @N values, we used *normalized Discounted Cumulative Gain* (*nDCG*) (Shani & Gunawardana, 2011). Details about evaluation methods and metrics used in this study are explained in Sections 3.4 and 3.5.

Following all predictions and tests in different sparsity levels, we observed that as sparsity increases, the nDCG value decreases for all @N values as shown in Fig. (3.7). For example; in the ML1M dataset, $\mathbf{Tr_2}$ was less sparse than its subset, $\mathbf{Tr_3}$. We could see that the $\mathbf{Tr_2}$'s nDCG result is better than the results of its subsets. As the length of the top-N recommendation list increases (it is @N in Fig. 3.7), the nDCG value rises, but as the sparsity expands, the nDCG values of all @N values do not converge successfully and the performance decreases.

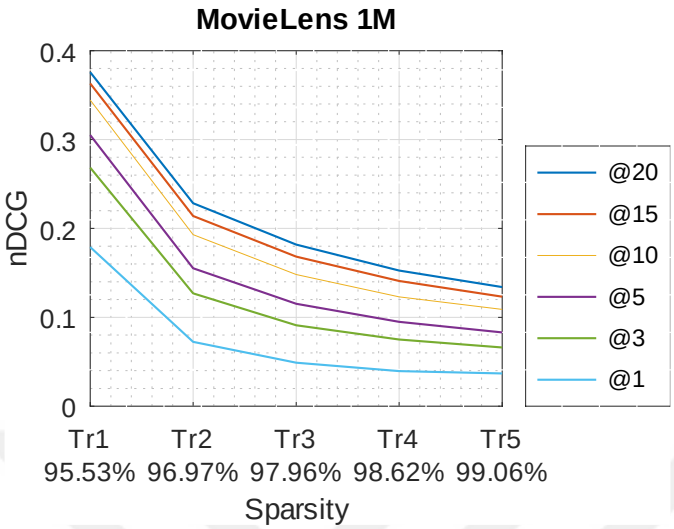

Figure 3.7. Performance of the PureSVD algorithm in different sparsity percentages of MovieLens 1M dataset evaluated with nDCG metric

In live recommender systems, present states of the user-item interactions should become the subspaces of future states. Based on the use of the PureSVD, we conclude that the impact of the latent factors of the rating matrix could be enhanced with the recent relations that can be created virtually. Therefore, in our new model, we aim to decrease sparsity to make better recommendations.

3.2.7. ImposeSVD Model Definition

In this study, we have developed a model to create new virtual connections in the referral system. With the help of the new predicted relations, system sparsity could be decreased and the recent success of algorithms may increase. When we estimate the impute matrix by using Eq. (3.20) with **K** created by the **z-scoREC** algorithm, we get the $\tilde{\mathbf{R}}$ predictions. Our goal is to impute users' missing ratings in the original rating matrix **R**. So, we removed previously rated items in **R** by applying Eq. (3.35) to obtain $\acute{\mathbf{R}}$. All we need was unrated items to scale the real rating values.

$$\acute{\mathbf{R}} = \tilde{\mathbf{R}} \odot \neg \mathbf{R} \tag{3.35}$$

In Eq. (3.35), $\neg\mathbf{R}$ is obtained by swapping all zeros and ones for their opposite from the binarized **R** matrix. Finally, $\acute{\mathbf{R}}$ represents new predicted values that users have not previously interacted with them. Decomposition of the latent factors in the $\acute{\mathbf{R}}$ matrix might have caused the existing real evaluations to worsen. Therefore, we did not apply any normalization process when calculating the **S** proximity matrix. The impute values obtained in the $\acute{\mathbf{R}}$ matrix might have been in a different range than the actual rating values. So, we applied row-max normalization before imputing the latest $\acute{\mathbf{R}}$ matrix. Because users could have different rating ranges in the impute matrix, we normalized each row by dividing its max value by the Eq. (3.36) where L_∞ was the norm in their row-vectors. Even small values in the $\acute{\mathbf{R}}$ were valuable, so we applied the exponential scale to make these values meaningful on the actual rating scale. The method we applied here is to degrade the difference between small and large values, as opposed to the process in Eq. (3.37). In this way, we have ensured that unpopular predictions are noticeably impacted.

$$\acute{\mathbf{R}} = \acute{\mathbf{R}} \,/\, \|\acute{\mathbf{R}}\|_\infty \tag{3.36}$$

$$\acute{\mathbf{R}} = \exp[-\mathbf{1}/\, \acute{\mathbf{R}}^T] \tag{3.37}$$

With these normalization processes, we compressed the values to be imputed into the (0,1) on the same range with the actual data. Then, we could merge our impute matrix with the original matrix as shown in Eq. (3.38);

$$\mathbf{R} = \mathbf{R} + \acute{\mathbf{R}} \tag{3.38}$$

After imputing the $\mathbf{R}$ matrix, we got the final impute matrix $\mathbf{Ŗ}$. Singular Value Decomposition of the $\mathbf{Ŗ}$ was estimated via Eq. (3.39)

$$\mathbf{Ř} \approx \mathbf{\bar{U}\Sigma\bar{Q}^T} \tag{3.39}$$

As we explained before, $\mathbf{\bar{U}}$ is user factors, $\mathbf{\bar{Q}}$ is item factors and $\mathbf{\Sigma}$ is singular values of decomposition.

3.2.8. Generating Predictions

Similar to Eq. (3.33), our prediction formula with $\mathbf{\bar{Q}}$ factors obtained from the $\mathbf{Ŗ}$ matrix was estimated as in Eq. (3.40)

$$\mathbf{\tilde{R}} \approx \mathbf{Ŗ\bar{Q}\bar{Q}^T} \tag{3.40}$$

The above equation differs from Eq. (3.33) because a row of the imposed matrix for the user as R_u is richer than the original user predictions. Algorithm 4 shows the basic steps of the proposed ImposeSVD method.

Algorithm 4: **ImposeSVD**

Input: Ratings Matrix $\mathbf{R} \in \mathbb{R}^{n*m}$, λ shrinkage value, f low rank size

Output: $\tilde{\mathbf{R}} \in \mathbb{R}^{n*m}$ final prediction matrix

// Explained in Algorithm 1, $O(n*m^2)$

1: $\mathbf{K} \leftarrow$ **z-scoREC**$(\mathbf{R}, \lambda)$

// Multiplying rating matrix with weight matrix, $O(n*m^2)$

2: $\tilde{\mathbf{R}} \leftarrow \mathbf{RK}$

// Hadamard product with nonzero elements of **R**, $O(n*m)$

3: $\acute{\mathbf{R}} \leftarrow \tilde{\mathbf{R}} \odot \neg\mathbf{R}$

// Row-based normalization by infinity-norm of $\acute{\mathbf{R}}$, $O(n*m)$

4: $\acute{\mathbf{R}} \leftarrow \acute{\mathbf{R}} / ||\acute{\mathbf{R}}||_\infty$

// Element-wise exponential scaling, $O(n*m)$

5: $\acute{\mathbf{R}} \leftarrow \exp[-1/\acute{\mathbf{R}}]$

// Impose Matrix; element-wise sum, $O(n*m)$

6: $\mathbf{Ŗ} \leftarrow \mathbf{R} + \acute{\mathbf{R}}$

// Decomposition of Ŗ with f rank. Also, could be estimated via *Golub-Kahan-Lanczos Bidiagonalization*, $O(\#nonzero(\mathbf{R}) * f)$

7: $\check{\mathbf{R}} \leftarrow \mathbf{U}_{n*f}\mathbf{\Sigma}_{f*f}\mathbf{Q}_{f*m}^{T}$

// Latent factors product for f rank, $O(n*m*f + n*f)$

8: $\tilde{\mathbf{R}} \leftarrow \mathbf{ŖQQ}^{T}$

9: **return** $\tilde{\mathbf{R}} \leftarrow \mathbf{ŖQQ}^{T}$

Figure 3.8. Pseudocode for ImposeSVD

As it can be seen from Algorithm 4, our method uses the imposed matrix **Ŗ**, which keeps the original relations in the **R** matrix, but imposes new relationships for the null entries in the original **R** matrix, to make predictions. Although our proposal is an SVD-based method, it is quite different from other SVD-based methods that are PureSVD, EigenREC, and HybridSVD:

i) PureSVD uses the original **R** matrix by changing null values to 0, however our method uses the imposed matrix **Ŗ** for predictions.

ii) EigenREC applies scaling to item similarity matrix obtained from **R** and changes all values in **R** matrix, however our method keeps non-null values in **R** but imposes new values for null entries in **R**, therefore we update **R** matrix differently from EigenREC.

iii) In our model, we use our z-scoREC method to compute the item similarity matrix, however EigenREC employs Pearson, Cosine, or Jaccard

similarity metrics and chooses the best metric for the dataset at hand to form the item similarity matrix. Therefore, we reduce the number of computations to be made with respect to EigenREC. iv) EigenREC uses the same prediction formula with the PureSVD that employs the **R** matrix. Our model, on the other hand, uses the imposed matrix $\mathbf{\tilde{R}}$ for making predictions.

v) Our model considers only user-item rating matrix **R** and does not require additional information about users or items, however HybridSVD incorporates side information of users or items in addition to CF.

3.2.9. Computational Complexity of ImposeSVD

The algorithm for imposing predicted prior values into the original matrix and the final prediction estimate is given in Algorithm 4. In Line 1, we estimated the $\mathbf{K}^{m*m}$ weight matrix between the items. As it can be seen in Algorithm 3, the overall time complexity is $O(n*m^2)$. In Line 2, we multiplied our novel weight matrix **K** with the original rating matrix and this process corresponds to the $O(n*m^2)$. $\mathbf{\tilde{R}}$ is our both priori predictions and **z-scoREC** predictions. With Line 3, we removed user-rated items from this prediction matrix because we only wanted to normalize the user's unrated predictions. This process is applicable to only nonzero elements of **R** where complexity is at most $O(n*m)$. In Line 4, we scaled the prediction matrix by dividing each row values by the row-max value. In Line 5, we normalized the ratings in an exponential scale. Both Line 4 and Line 5 correspond to $O(n*m)$. In Line 6, we merged the initial **R** matrix with the imposing matrix in $O(n*m)$ time. Line 7 is the decomposition phase of the $\mathbf{\tilde{R}}$, which is our imposed matrix. Golub and Van Loan (2013) showed how to efficiently compute the SVD of a sparse matrix $\mathbf{R} \in \boldsymbol{R}^{n*m}$ $(n>m)$ which could be applied in line 7. This computation method of Golub and Kahan (2013) is based on bidiagonal factorization and it is efficient in sparse matrices. For a large matrix, SVD could be estimated via *Golub-Kahan-Lanczos Bidiagonalization* with $O(\#nonzero(\boldsymbol{R}) * f)$, where $\#nonzero(\boldsymbol{R})$ is the number of nonzero elements in **R** which is less than $n*m$. In line 8, we estimated $\mathbf{\tilde{R}}$ with $\mathbf{\tilde{R}}$

impose matrix and item factor **Q** from decomposition. This process depends on a low *f* rank value and corresponds to the $O(n*m*f + n*f)$. As *f* is much smaller than *n* and *m*, time complexity of the algorithm is $O(n*m^2)$.

3.3. Datasets

To evaluate the recommendation quality and performance of our algorithms, we used six datasets from various domains, which are MovieLens 1M (Harper & Konstan, 2015), MovieLens 10M (Harper & Konstan, 2015) and Netflix (Bennett & Lannning, 2007) datasets for movie recommendations, R2-Yahoo! Music dataset for song recommendations (Yahoo, 2020), BookCrossing for book purchases (Ziegler et.al., 2005), and implicit data crawled from an often-cited paper about Pinterest for image recommendations (Geng et.al., 2015; He et.al., 2017). User, item, and rating counts of the datasets with their sparsity and density values are shown in Table 3.1, where sparsity percentage is calculated as **(1-density) *100**, in which density formula is **density = #ratings / (#users x #items)**.

Table 3.1. Evaluation Datasets

	ORIGINAL DATASETS			
Dataset	**# User**	**#Item**	**#Rating**	**Sparsity**
MovieLens 1M	6040	3952	1M	95.809
MovieLens 10M	72 K	10681	10M	98.692
Netflix	480 K	17770	100M	98.822
R2 -Yahoo! Music	1.8 M	136 K	717 M	99.707
BookCrossing	246.7 K	255.7 K	716109	99.995
Pinterest Image	46 K	882 K	2.6M	99.993

MovieLens (ML): MovieLens is a popular dataset in the Recommender Systems literature which is first released in 1998, describe people's expressed

preferences for movies. This dataset contains movie ratings from the online movie recommender service MovieLens. In this thesis we used "1M" and "10M" versions.

Netflix Prize Dataset: The Netflix Prize competition was held by Netflix in 2009, and the grand prize of US $1,000,000 was given to the best recommendation algorithm. Netflix released a dataset containing 100 million anonymous movie ratings for competitors (researchers) that could beat the accuracy of its recommendation system (Cinematch).

Yahoo Webscope Dataset: R2 -Yahoo! Music dataset represents a collection of the Yahoo! Music community's preferences for various musical artists. This huge dataset contains over ten million ratings of musical artists given by Yahoo! Music users over the course of one month sometime before March 2004.

BookCrossing: The BookCrossing (BX) dataset was collected by Cai-Nicolas Ziegler from the BookCrossing community. It contains 255,7k users (anonymized but with demographic information) providing 1,149,780 ratings (explicit / implicit) about 271,379 books. We used implicit ratings of the BX in this thesis.

Pinterest Image Dataset: He et al. (2017) constructed this dataset from paper data that were constructed by (Geng et.al., 2015) for evaluating content-based image recommendation. The original dataset is huge but highly sparse and its details are given Table 4.1. He et al. (2017) filtered the dataset that retained only users with at least 20 interactions (pins). This dataset contains 55,187 users and 1, 500, 809 interactions (pins). Each interaction denotes whether the user has pinned the image to her own board.

Selected datasets are well-known recommendation datasets that are used in most of the previous studies in this field. Attributes of the datasets are given in Table 3.1.

Because of the huge size of these datasets, we created particular subsets by which we evaluated the benchmarks. In this way, benchmarks and parameter tunings

were estimated much faster. We obtained subsets similar to the subset sampling methods used in the related studies (Kabbur et.al., 2013; Ning & Karypis, 2011). The sampling methods were run recursively until the specified conditions were met for each dataset. With these sampling methods, we aimed to create a subset that is close to their original densities. Finally, we normalized the ratings of the non-binary datasets by dividing each one by the maximum rating value in the dataset.

3.4. Evaluation Metrics

In this thesis, we didn't consider the similarity of the estimated ratings with actual ratings; instead, we measured the quality of the items, which are recommended to the users. To evaluate the top-N recommendation quality of these lists, we adopted Cremonesi et al. (2010) method for the benchmark algorithms. Following this method, we created a list by randomly choosing 1000 unrated items of the active 'test user' in addition to the test item. As a result, we obtained 1001 items that the active test user had not seen before. Later, these 1001 items were sorted based on their prediction scores, which were estimated by prediction algorithms. N items were obtained out of the 1001-item list as a result of their cut-off higher prediction scores. This final list is the top-N item recommendation list for the 'test user'. In our experiments, we used several values for N that are 1, 3, 5, 10, 15, and 20 items for the length of recommendation lists.

In AcoRec with a change, instead of selecting 1000 items, we decided to calculate the top-N lists by sorting all the items that the user did not click on. This is a more difficult challenge but yielded a more consistent result. Because AcoRec is a probabilistic model, randomly selected elements here could vary the results significantly. Later, all unclicked items were sorted based on their prediction scores, which were estimated by prediction algorithms.

Evaluation of predictions and recommendations is an important progress of the Recommender System studies. Recommender Systems require quality measures and evaluation metrics to know the quality of the techniques, methods and

algorithms. Because of evaluation measures, Recommender System recommendations have gradually been tested and improved.

Herlocker et al. (2004) classifies recommendation accuracy metrics into three classes: *predictive accuracy* metrics, *classification accuracy* metrics, and *rank accuracy* metrics.

- Predictive accuracy metrics measure how close the recommender system's predicted ratings and rankings are to the true user ratings and rankings. (Accuracy of Estimated Rating examples are MAE, NMAE, RMSE. Accuracy of Estimated Ranking metrics are Pearson, Spearman's rank, NDMP)
- Classification metrics measure the frequency with which a recommender system makes correct or incorrect decisions about whether an item is good. (Precision, Recall, F1 Measures, ROC Curves, AUC, Hit-rate are some these metrics)
- Rank accuracy metrics measure the ability of a recommendation algorithm to produce a recommended ordering of items that matches how the user would have ordered the same items. (Mean reciprocal rank (MMR), Average reciprocal hit rank (ARHR), nDCG, Half-life Utility Metric etc. are the example metrics)

We used utility-based metrics including *Hit Rank* also *Recall* (Deshpande & Karypis, 2004), *normalized Discounted Cumulative Gain* (Shani & Gunawardana, 2011)**,** *Coverage* (Herlocker et.al., 2004) and *R-Score (Half-Life)* (Shani & Gunawardana, 2011) to measure the quality of the items in the lists in terms of their relevancy to the user. These metrics are explained below with their formulas. In these formulas; ***T*** is the count of the test items in 'test set', ***N*** is the length of the recommendation list and ***i*** is the position of the recommended item in the list. If the

ranked ***item(i)*** in the list belongs to the test user from 'test set', we called this item a *'relevance item'* for the user and set ***rel(i)*=1**, if it is not we set ***rel(i)*=0** for the test user.

3.4.1. Hit Rank (HR)

To evaluate the recall score of the algorithm for specific datasets in different list lengths, we divide the sum of all '*relevance items'* by the number of items in the test set. The *Hit Rank* formula is given in Eq. (3.41).

$$HR(@N) = \frac{1}{T}\sum_{i=1}^{N} rel(i) \tag{3.41}$$

3.4.2. Normalized Discounted Cumulative Gain (nDCG)

The position of the '*relevance item'* in the lists is ignored in the HR. The recommendations at the top of the list are more valuable than others. So, we measured the importance of the position of the item in the list by the ratio of the *'relevance item'* to its position in the list. nDCG gives importance to the gain of the position logarithmically while considering the list quality at the same time. In this nDCG metric; firstly, *Discounted Cumulative Gain (DCG)* of the test set was estimated as in Eq. (3.42) and then *Ideal Discounted Cumulative Gain (IDCG)* was estimated as in Eq. (3.43) for every test item in the top-N list. And then we normalized these gain values with Eq. (3.44) and obtained the *nDCG* value for a benchmark test.

$$DCG(@N) = \frac{1}{T}\sum_{i=1}^{N} \frac{rel(i)}{log_2(i+2)} \tag{3.42}$$

$$IDCG(@N) = \frac{1}{T}\sum_{i=1}^{N}\frac{1}{log_2(i+2)} \tag{3.43}$$

$$nDCG(@N) = \frac{DCG(@N)}{IDCG(@N)} \tag{3.44}$$

3.4.3. R-Score (Rs)

R-Score considers that the probability of selecting a relevant item in the top-N list that goes down exponentially (Shani & Gunawardana, 2011). The R-Score formula is given in Eq. (3.45). Different from other metrics, the parameter **α** specifies the slope of the decay curve and exhibits scroll or discovery of users for a recommendation list. A higher **α** value indicates patient users.

$$Rs(\alpha) = \frac{1}{T}\sum_{i=1}^{T}\frac{rel(i)}{2^{\frac{i-1}{\alpha-1}}} \tag{3.45}$$

3.4.4. Coverage

The coverage metric measures the ability of a recommender system with the percentage of different elements in total items in the whole recommendation list. We define the coverage of the system as the average of the user's coverage in Eq. (3.46)

$$Coverage(@N) = \sum_{i=1}^{\#U}\frac{U_{uen}i}{|I|} \tag{3.46}$$

where $U_{uen}i$ is the number of different items in the recommended list, and $|I|$ is the number of items counted in the system.

3.5. Evaluation Methods

3.5.1. AcoRec

In the AcoRec model, we only used binary rating values. We used three datasets for AcoRec which are Movie-Lens 1M (ML-1M)**,** Netflix datasets for movie recommendations, and Yahoo! R2-Music dataset about music recommendations. Attributes of the datasets are given in Table 3.2. For ML-1M, 4 and 5-star ratings were converted to binary one while others were converted to zero. After that process, we selected the users who listened to at least one item and selected the movies which were rated by at least one user, and in this way, we got a very sparse dataset than the original. Due to its large size, in the R2-Yahoo! Music dataset, 10% of the ratings were taken from the first CSV file and 4 and 5-star ratings were converted to binary one while others were converted to zero. We selected the users who listened to at least 20 and at most 250 songs and selected the songs, which were listened to by 20 to 250 users. In the Netflix dataset, we selected the small public sample of the original from the Cornac[1] repository.

Table 3.2. Sampled Datasets for AcoRec

	SAMPLED SUBSETS FOR AcoRec					
Dataset	**Domain**	**#User**	**#Item**	**#Ratings**	**Sparsity**	**Density**
ML-1M	Movie	6038	3533	575281	97.302	2.698
Netflix	Movie	8324	2679	366432	98.488	1.512
Yahoo! R-2	Music	5357	5627	202042	99.330	0.670

We adopt the *k-fold cross-validation* method for splitting raw datasets to evaluate the algorithms. We shuffled all datasets randomly and then split them into *k*=5 sampled datasets. For each unique sampled group, we take it as a probe set and hold out this ‘probe set’ from the raw dataset. We called raw datasets ‘training set’

[1] https://github.com/CornacAI

after removing ‘probe set’ from it sequentially. From these probe sets, we selected users and their ratings who met the criteria according to the scenarios that we explained in the experiments. These selected users and their ratings in the ‘probe set’ are called the ‘test set’. In this way, results for different users and items in each experiment completed an average estimate for us.

3.5.2. ImposeSVD and z-scoREC

In the z-scoREC and ImposeSVD models, we used both binary and scalar rating values. In MovieLens 10M dataset**,** we selected the users who rated between 20 and 500 items, and items that were rated by between 20 and 500 users. In the Netflix dataset**,** we selected the users who rated between 10 and 500 items and items, which were rated by between 5 and 250 users. Due to its large size, in the R2-Yahoo! Music dataset, 10% of the ratings were taken from the first CSV file and 5-star ratings were converted to binary one while others were converted to zero. We selected the users who listened to at least 10 and at most 200 songs and selected the songs, which were listened to by 20 to 200 users. Implicit ratings taken from the BX dataset included the users who had at least 10 purchases and books that were bought by at least 10 users. In the Pinterest dataset, we transposed the dataset from a perspective of board recommendations for images to provide meta-information of boards for the HybridSVD method. As a result, we selected the images that were pinned at least in 10 boards and boards that had at least 10 images. Attributes of the used datasets are given in Table 3.3.

Table 3.3. Sampled Datasets for ImposeSVD and z-scoREC

	SAMPLED SUBSETS FOR ImposeSVD AND z-scoREC					
Dataset	**Domain**	**#User**	**#Item**	**#Ratings**	**Sparsity**	**Density**
MovieLens 1M	Movie	6040	3952	1000209	95.809	4.191
MovieLens 10M	Movie	4101	2931	144453	98.798	1.202
Netflix	Movie	7249	5548	131268	99.673	0.327
R2 -Yahoo! Music	Music	7456	5047	182426	99.515	0.485
BookCrossing	Book	2617	3871	87849	99.132	0.868
Pinterest Image	Image	3862	4996	85805	99.555	0.445

We adopt the *holdout* method for splitting raw datasets to evaluate the algorithms. First, we created the *out-of-sample* that we called a probe set for each dataset. Then, we randomly selected 1.4% of the ratings in the raw datasets and removed this 'probe set' from the raw datasets. We called raw datasets 'training set' after removing 'probe set' out of it. From these probe sets; we selected random users and their ratings that met the criteria according to the scenarios that we explained in the experiments. These selected users and their ratings in the 'probe set' are called 'test set'. Because of the random selections in 'probe set', we created at least ten *repeated holdout evaluations* for 'training set' and 'test set' to evaluate the majority of the dataset. In this way, results for different users and items in each experiment produced an average estimate for us.

4. RESULTS AND DISCUSSION

4.1. AcoRec Evaluation Results

To evaluate the performance of our meta-heuristic algorithm, we evaluated our experiments in two scenarios. The first scenario is built to see how accurate our algorithm is in assessing the recommendations for cold-start users, and the second is to measure the long-tail item diversity in recommendations.

We show the best percentage value for the Coverage value related to the best nDCG parameters for each algorithm. We considered it a fairer way of evaluating the diversity of items on that list.

4.1.1. Selected Benchmark Algorithms for AcoRec and Parameter Tunings

For the benchmark tests of AcoRec, we used the three item-based similarity models as input of our approach which are Gram-matrix, Cosine Similarity, and Jaccard Similarity.

Base-Gram, Base-Jaccard, Base-Cosine: The item-based baseline models are estimated by Eq.(3.17), Eq.(3.18), and Eq.(3.19)**.**

TARS: This is a state-of-the-art ACO model in recommender systems. It offers a user-based model that creates a trust-based user relationship graph, detects similar users, and makes a rating estimation (Bedi & Sharma, 2012).

RP^{3}ß: A random walk model that recommends based on the user-item graph with extending diversification that eliminates tendency on popular items (Paudel et.al., 2016).

RecWalk^{PR}, RecWalk^{K}: Random-walk-based methods to capture new rich network interactions for top-N recommendation lists (Nikolakopoulos & Karypis, 2019).

SLIM: A well-known item-based CF method building a sparse coefficient item model L_1-*norm* and L_2-*norm* on the rating matrix (Ning & Karypis, 2011).

EASE^R: A robust linear model that shows the closed-form solution of Ridge Regression in a manner of vanilla auto-encoders (Steck, 2019).

UCF: Resnick's user-based CF approach. We used Pearson Similarity for obtaining user similarities (Resnick et.al., 2004).

Random: The baseline method that we evaluate in benchmarks with filling empty cells in the user-item matrix with random values between (0,1).

Popular: A baseline algorithm that evaluates items according to their usage frequency.

Parameter Tunings of Algorithms: TARS method is evaluated between [10-250] values in 10 steps for the (k) user neighbor size and confidence values between [0,1] range with 0.1 steps. RP$^3\beta$ algorithms are tested between [0,2] beta and alpha (β,α) values with 0.05 steps. SLIM algorithm is tested alpha with {0.01, 0.05, 0.1, 0.5, 2, 5} values and beta with { 1e-4, 1e-3, 5e-3, 5e-2, 0.1, 0.2, 0.5, 1.0} values. To execute SLIM, we took the standard Elastic-Net implementation provided by the sci-kit learn package for Python. EASE^R method is evaluated between [10-100] values in 10 steps and [500-20000] values in 500 steps for the (λ) value. For RecWalk^{PR} and RecWalk^K models, we evaluated both Cosine and SLIM as input models like in the original paper and chose the best model for every benchmark. Our AcoRec method is evaluated between [1...250] values for ant size (archive-size) and [1...100] values for iteration count. Due to the random choices, each experiment for AcoRec is repeated 10 times, results are averaged and the best parameter results are chosen while creating transition probabilities during iterations. In all scenarios, we accomplished Grid-Search to find the best parameters working together in each algorithm. In the Section 4.1, we have shown the best results obtained by the best parameters for each algorithm.

4.1.2. Cold-start user scenario

Cold-start users have fewer ratings on the system, so it is more difficult to give quality recommendations to them (Son, 2012; Bobadilla et.al., 2016). To

evaluate our algorithm for cold-start users, we obtained heat or warm users as candidate users from the probe set who was also in the training set. We formed them as cold-start users by decreasing their rating counts in the training set and we selected randomly 100 users from the probe set who had at least one full rating in the probe set and at least twenty ratings in the training set. In the evaluation process, some studies put three items in the training set to define cold-start users (Son, 2012), some studies received 5% of the user's rates (Nikolakopoulos et.al., 2019), and some other studies tested both in numbers ranging from (1...20) or used percentage rates (Ahn, 2016). For a harder challenge, we kept random ratings that were between 5 and 10 of the particular users in the training set and other ratings of these users were removed from the training set. Consequently, this process transformed candidate users into cold-start users, represented by a minimum of 5 and a maximum of 10 random ratings in the training set. Just like a real scenario, the random ratings of these users could be lower ratings or higher ratings in different distributions.

Table 4.1. Comparisons of the algorithms in a cold-start user scenario

	MovieLens 1M			Netflix			Yahoo! R-2		
@10	HR	nDCG	Cov.	HR	nDCG	Cov.	HR	nDCG	Cov.
Popular	0.057	0.025	0.36	0.078	0.037	0.50	0.002	0.001	0.21
Random	0.003	0.001	**25.12**	0.010	0.005	**31.05**	0.000	0.000	**20.61**
Base^{Gram}	0.080	0.039	1.00	0.101	0.048	0.87	0.073	0.036	10.85
Base^{Cosine}	0.092	0.042	4.30	0.107	0.053	3.98	0.071	0.035	14.25
Base^{Jaccard}	0.086	0.039	6.34	0.108	0.054	6.23	0.073	0.033	14.25
UCF^{Pearson}	0.100	0.044	4.36	0.127	0.060	3.28	0.090	0.040	12.77
TARS	0.099	0.047	3.88	0.132	0.065	3.58	0.081	0.037	11.69
RecWalk^{K}	0.100	0.044	8.62	0.136	0.065	3.54	0.089	0.043	12.56
RecWalk^{PR}	0.094	0.044	8.75	0.129	0.061	3.67	0.092	0.044	13.68
SLIM	0.100	0.044	8.62	0.136	0.065	3.54	0.089	0.043	12.56
EASE^{R}	0.099	0.045	10.09	0.138	0.065	2.81	0.090	0.039	14.56
RP3$^{\beta}$	0.107	0.049	3.60	0.145	0.067	6.36	0.087	0.042	14.17
AcoRec^{Gram}	0.118	0.054	**10.37**	0.143	0.066	**11.87**	**0.097**	**0.048**	**16.18**
AcoRec^{Cosine}	**0.121**	**0.057**	8.31	**0.162**	**0.076**	8.76	0.080	0.039	15.52
AcoRec^{Jaccard}	0.104	0.047	10.32	0.129	0.062	9.23	0.073	0.034	15.64

Experimental results based on the HR, nDCG, and Coverage metrics are summarized in Table 4.1. When we evaluated AcoRec in the cold-start user scenario; we observed that our AcoRec outperformed most other algorithms in all datasets.

In addition to building up the quality list, we observed that AcoRec implementations outperformed all other algorithms in the Coverage metric, which measures the diversification of the items on the list. The Baseline Method on which we base the AcoRec algorithm provided an improvement in Gram Matrix and nDCG measurement in all datasets. As stated by Dacrema et al. (2019), an algorithm in

which parameters are tuned well can outperform many deep learning algorithms. Our study confirms this argument as we fairly evaluated algorithms and obtained successful results in different scenarios.

While the similarity matrices we use as inputs fail against other algorithms when used as predictors alone, our model is successful in all models when it was integrated with ACO.

Another observation is that although the Coverage percentages of the Random algorithm are on the top, the HR and NDCG values are close to zero. In contrast to this situation, the most notable feature of our algorithm is that not only the Coverage percentage is high in the lists recommended to users but also the quality of the lists is also high.

We also observe that whereas the Cosine item-similarity model is more successful in MovieLens and Netflix datasets, the Gram matrix is more successful in terms of list diversity in cold-start recommendations. The Yahoo dataset is a sparser dataset than the other experimental sets used in this study, so the diversity in the lists shown to the users in this dataset is greater in all algorithms. The results of the analysis revealed that the Gram matrix is quite successful in the Yahoo dataset, too. Moreover, we observed that the $RP3^{\beta}$ algorithm is a very successful algorithm with correct parameter tuning.

AcoRec takes an item-based similarity model as input. If we compare our three AcoRec model results with their own input item-based similarity model results (Table 4.1), we estimated compared percentage results for each metric. Values on Table 4.2 show the improvement percentage of each AcoRec model on its base item-similarity model. The results show that our AcoRec models produce significantly improve their base input models. In particular, the improvement in Gram-matrix and Jaccard similarity models is better than the improvement in the Cosine similarity model.

Table 4.2. Comparisons of AcoRec with its base algorithms in a cold-start user scenario

	MovieLens 1M			Netflix			Yahoo! R-2		
@10	HR	nDCG	Cov.	HR	nDCG	Cov.	HR	nDCG	Cov.
AcoRec Gram	29.3	27.8	90.3	28.9	27.2	92.7	25.1	25.3	28.2
AcoRec Cosine	18.9	19.7	36.5	16.2	12.9	32.5	1.0	2.8	2.7
AcoRec Jaccard	23.8	26.2	48.2	34.0	30.9	54.6	10.7	9.4	8.1

4.1.3. Long-tail items scenario

Popular items are familiar to users, and thus, recommending these items might be boring for users (Anderson, 2006). Therefore, recommending unpopular items has always been more attractive. Traditional CF algorithms dealing with relations between popular items or popular users overshadow diverse relationships. Considering that the quality of models depends on the diversity of recommendations they offer, these CF methods might be unable to generate a diverse range of suggestions in the datasets especially when the data is inadequate (Yin et.al., 2006). The diversity can be achieved when some unpopular items are recommended to the users. Based on these arguments, we also evaluated the reaction of our algorithms for long-tail item recommendations. To obtain an experiment environment suitable for the long-tail scenario, we adopted the method in (Cremonesi et.al., 2010). As observed by the authors in the study (Cremonesi et.al., 2010), the most popular 1.7% items represent 33% of the ratings included in the Netflix dataset and they called these 1.7% items as short-head items while the remaining items are called long-tail items. Following this evaluation method (Cremonesi et.al., 2010), we sorted the items in the dataset according to their popularity to evaluate the existence of long-tail items in the recommendation lists. In doing so, we determined the items' popularity by their rating frequency and sorted them in descending order by the number of ratings they had. On the sorted item list according to their frequency, from

top to bottom, we marked the items as ‘short-head’ items until the sum of the item frequencies equal to or higher than the 33% of the total ratings and marked the remaining as ‘long-tail’ items. In the probe set, we kept ‘long-tail’ items and removed the others. We created a ‘test set’ out of the probe set by random selection of 250 users who gave at least one full rating to ‘long-tail’ items. As a result, we randomly selected users with unpopular tastes for each repeated holdout evaluation.

According to the results presented in Table 4.3., we observed that our algorithm outperforms all algorithms in all datasets. Because of the creation of a long-tail scenario, the Popular baseline algorithm validates our dataset as expected, counts zero results, and fails against the Random algorithm. The performance measure values of our algorithm against other algorithms are better in the long-tail scenario than in the cold-start scenario. But our algorithm resulted in a higher difference and percentage in outperforming all algorithms in this scenario.

Table 4.3. Comparisons of the algorithms in long-tail item recommendation scenario

	MovieLens 1M			Netflix			Yahoo! R-2		
@10	HR	nDCG	Cov.	HR	nDCG	Cov.	HR	nDCG	Cov.
Popular	0.000	0.000	0.00	0.000	0.000	0.000	0.000	0.000	0.00
Random	0.004	0.002	51.22	0.001	0.000	60.66	0.004	0.002	35.89
Base^{Gram}	0.000	0.000	0.00	0.000	0.000	1.67	0.055	0.024	15.71
Base^{Cosine}	0.009	0.003	4.29	0.014	0.008	5.60	0.130	0.062	24.47
Base^{Jaccard}	0.016	0.008	6.33	0.030	0.015	7.27	0.143	0.068	24.84
UCF^{Pearson}	0.038	0.018	14.94	0.037	0.016	14.99	0.112	0.052	28.53
TARS	0.029	0.013	10.77	0.032	0.015	8.00	0.097	0.046	24.95
RecWalk^{K}	0.070	0.032	13.29	0.086	0.041	11.62	0.138	0.063	26.32
RecWalk^{PR}	0.067	0.030	12.90	0.078	0.037	11.13	0.136	0.063	25.59
SLIM	0.071	0.033	13.41	0.089	0.043	11.14	0.133	0.061	27.04
EASE^{R}	0.064	0.028	13.89	0.092	0.045	11.96	0.124	0.056	30.12
RP3$^{\beta}$	0.115	0.053	16.79	0.132	0.063	27.36	0.195	0.094	27.94
AcoRec^{Gram}	0.145	0.068	23.19	**0.195**	0.095	37.12	0.198	0.095	31.92
AcoRec^{Cosine}	0.160	0.075	21.73	0.184	0.089	20.66	0.177	0.085	31.66
AcoRec^{Jaccard}	0.173	0.083	**25.45**	0.184	0.090	17.39	0.164	0.080	29.07

As stated before AcoRec takes an item-based similarity model as input. If we compare our three AcoRec model results with their own input item-based similarity model results (Table 4.3), we estimated compared percentage results for each metric. Values on Table 4.4 show the improvement percentage of each AcoRec model on its base item-similarity model. The results show that our AcoRec models produce significantly improve their base input models in the long-tail scenario also. The improvements here are better than the cold-start users' results, showing that the algorithm is particularly successful in highlighting different products.

Table 4. 4 Comparisons of AcoRec with its base input algorithms in a long-tail item scenario

	MovieLens 1M			Netflix			Yahoo! R-2		
@10	HR	nDCG	Cov.	HR	nDCG	Cov.	HR	nDCG	Cov.
AcoRec Gram	100.0	100.0	100.0	99.9	99.9	95.5	72.4	74.5	50.1
AcoRec Cosine	91.0	90.8	75.1	83.9	83.2	58.2	12.6	14.8	14.5
AcoRec Jaccard	94.5	95.7	80.3	92.1	91.2	72.9	26.5	27.5	23.4

4.1.4. Effect of the ant size & iteration count

The number of iterations is an important parameter in ACO algorithms, but when number of iterations is increased in ACO algorithms, the algorithm is usually slow. Figures 4.1 and 4.2 show the heatmaps of our algorithms showing the relationship between the number of ants and the number of iterations in different input models and different scenarios. In Figures 4.1 and 4.2 the nDCG@10 was used as the quality measurement metric. The horizontal x-axis shows the number of iterations, and the vertical y-axis shows the number of ants. One of the outstanding features of our study is that it has fast convergence and that the cases of stagnation are lowly due to the structure of the algorithm. While examining the effect of the number of iterations in our experiments, we observed that it reached a high success value quickly and after this point, the success of the algorithm did not change with the increase in the number of iterations.

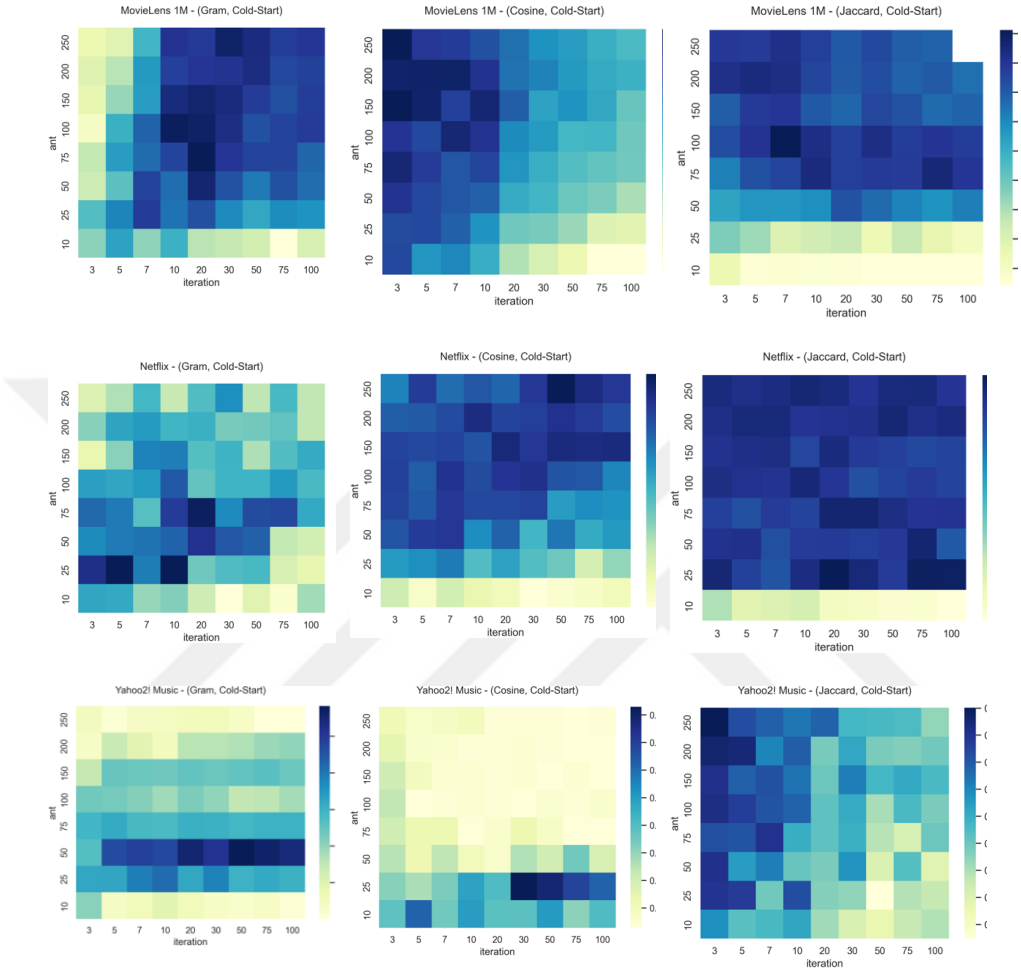

Figure 4.1. In the cold-start user scenario, the effect of the ant-size and iteration count is evaluated with the nDCG metric

In our experiments to investigate the effect of the number of ants on the achievement of the algorithm, we found out that the number of ants had a better effect than the iteration on the success of the algorithm. Regarding the effect of Gaussian Distribution at the time of training, we discovered an equal distribution in all localities. As the focus space tightened throughout the iterations, the ants came

out to meet at the same distribution position. At this point, when the variance decreased below a specified threshold rate, our algorithm completed its training. Figure 4.1 and 4.2 indicates the progress of the algorithm conferring to the increase in the number of ants.

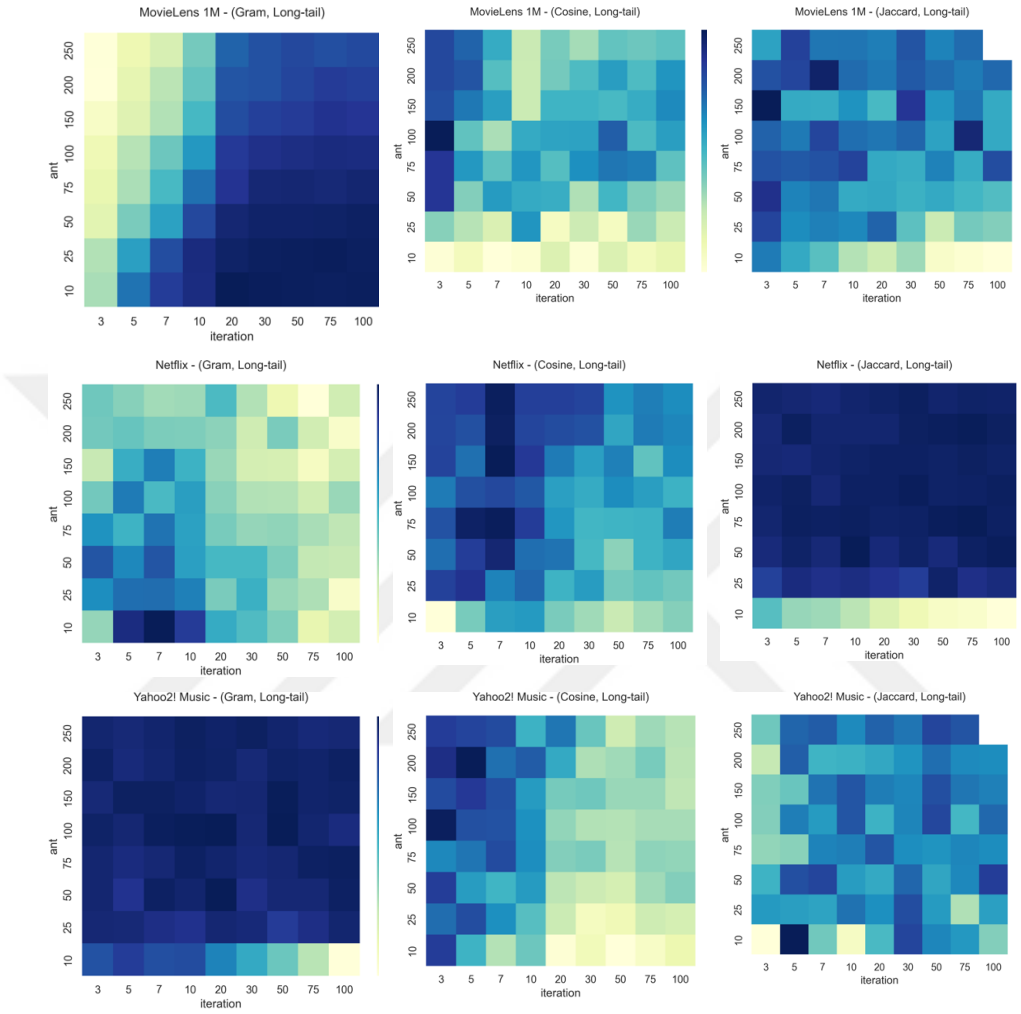

Figure 4.2 In the long-tail item scenario, the effect of the ant-size and iteration count is evaluated with the nDCG metric

4.1.5. Execution time of AcoRec

In this experiment, we evaluated the computational performance of our method in a hyperthreading test environment. The experiments were performed on the **TUBITAK ULAKBIM, High Performance and Grid Computing Center (TRUBA resources)**. TÜBİTAK ULAKBİM High Performance and Grid Computing Center is a national center providing high-performance computing and data storage for all research institutions and researchers in Turkey. TUBITAK ULAKBIM High Performance and Grid Computing Center, which started its operations in 2003, is included in TRUBA. Today, TRUBA serves our researchers with ~ 17,500 processor cores, 80 GPUs, and a 4PByte Luster distributed file system. We implemented AcoRec in GNU Octave (GNU Octave, 2022) and used a parallel package that is part of the Octave Forge project**.** We evaluated the same 100 users in every experiment and made multi-threading tests. The results are given in Figure 4.3 with execution time of core size. As you can see in the experiment results, AcoRec computational time decreases with the parallelable architecture. This is one of the main advantages of AcoRec against the other ACO-based RS implementations. AcoRec uses low-rank vectors and made personalized predictions for every user.

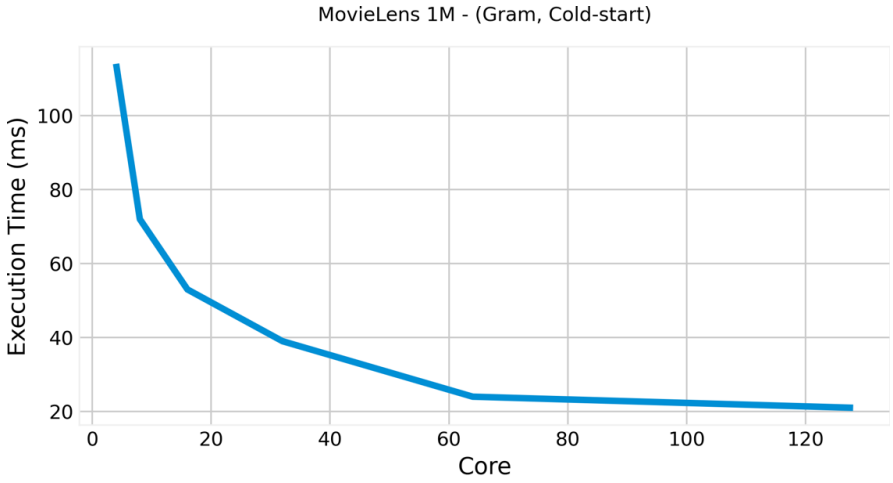

Figure 4.3. In the cold-start user scenario the execution time of our algorithm under different cores

4.2. z-scoREC and ImposeSVD evaluation and results

To measure the performance of our algorithms, we evaluated our experiments in three different scenarios. The first scenario was set to find out how accurate our algorithms were when assessing the recommendations for cold-start users, the second scenario was to assess the aspect of recommendations when the recommender system was fresh, and the third scenario was set to measure the long-tail item diversity in recommendations.

4.2.1. Selected Benchmark Algorithms for z-scoREC, ImposeSVD and Parameter Tunings

We compared **ImposeSVD and z-scoREC** against the three similar SVD-based top-N recommendation algorithms including **PureSVD** (Cremonesi et.al. 2010), **EigenREC** (Nikolakopoulos et.al., 2017; Nikolakopoulos et.al., 2019)**,** and **HybridSVD (**Frolov & Oseledets, 2019), which were explained in the previous sections in detail. In addition to these, we compared our algorithm with two popular item-based methods (i.e. item *k-NN* and Item-based collaborative filtering), and two other baseline methods (i.e., random and popular), which are explained below:

ItemKNN: The item-based model developed by Deshpande and Karypis (2004) is utilized as a benchmark model by many studies in the literature. In their original study, Deshpande et al. estimated item similarities via *Cosine Similarity* or *Probability Selection*. In the second step, they selected *k* similar items only by ignoring the other items. With this simplified item similarity matrix, they estimated the prediction matrix by equitation similar to Eq. (3.20)**.**

Item-Based Collaborative Filtering (ICF): Sarwar et al.*'s* (2001) item-based model is based on calculating the correlation between items and estimating scores with item-based prediction rules. In this method similar to the ItemKNN, a similarity between items is calculated by Cosine-based, Correlation-based, or Adjusted Cosine Similarity. Then the model tries to capture items that are similar to

users' liked items with prediction formulas Sarwar *et.al.* (2001). We preferred to use Adjusted Cosine Similarity for its efficiency in calculating better results.

Random: In addition to the item-based models, we also compared our algorithm with two baseline methods one of which is Random. With the help of the Random method, we evaluated the benchmarks by filling empty cells in the user-item matrix with random values between [0,1].

Popular: The second baseline method with which we compared our algorithm is Popular, which evaluates the items according to their frequency of use.

Parameter Tunings of Algorithms: All SVD-based algorithms were tested between [1-2000] factors (*f*) with 10 steps. EigenREC and HybridSVD were scaled with (*d)* parameters in the [-2,2] range with 0.05 steps. ItemKNN and ICF methods were evaluated between [10-250] values in 10 steps for the (*k*) item neighbor size. We evaluated our algorithms for lambda λ shrinkage values between [-1,1] range with 0.05 steps and tested all algorithms on each data set with the combinations of their parameters. Because of the random choices while creating 'probe sets', 'test sets', and 'random, 1001 items', we seeded random choices with the same seed number at the same stages for every method so that they were fairly evaluated in every *repeated holdout evaluation*.

In all scenarios, we accomplished Grid-Search to find the best parameters working together in each algorithm. In the Results section, we have shown the best results obtained by the best parameters for each algorithm. We presented the Grid-Search parameter tuning results in the Appendix B.

Similar to the implementations on SVD-based models, we also applied a method to determine the best *f* values in the HybridSVD **(**Frolov & Oseledets, 2019). In SVD-based models, in each experiment for an algorithm decomposition was calculated once and f_{max} value as 2000 was obtained for the matrix rank size of latent factors. Then we truncated the rank of latent factors with the evaluation steps between [1-2000] and evaluated all *f* values in each experiment. After all *repeated*

holdout evaluations, we averaged the results of all *f* values and selected the best *f* value result for each metric. This method reduced the search cost for SVD-based models, especially in Grid-Search **(**Frolov & Oseledets, 2019)

4.2.2. Cold-start user scenario

Cold-start users have fewer ratings on the system, so it is more difficult to give quality recommendations to them (Son, 2012; Bobadilla et.al., 2016). To evaluate our algorithm for cold-start users, we obtained heat or warm users as candidate users from the probe set who was also in the training set. We formed them as cold-start users by decreasing their rating counts in the training set and we selected randomly 100 users from the probe set who had at least one full rating in the probe set and at least twenty ratings in the training set. In the evaluation process, some studies put three items in the training set to define cold-start users (Son, 2012), some studies received 5% of the user's rates (Nikolakopoulos et.al., 2019), and some other studies tested both in numbers ranging from (1...20) or used percentage rates (Ahn, 2016). For a harder challenge, we kept random ratings that were between 5 and 10 of the particular users in the training set and other ratings of these users were removed from the training set. Consequently, this process transformed candidate users into cold-start users, represented by a minimum of 5 and a maximum of 10 random ratings in the training set. Just like a real scenario, the random ratings of these users could be lower ratings or higher ratings in different distributions.

Experimental results that are based on nDCG are shown in Figure 4.4. When we evaluated our algorithms in the cold-start user scenario; we observed that ImposeSVD outperformed other algorithms in all datasets and in all top-N variations. ImposeSVD is based on enhancing the PureSVD under sparse datasets and we observed that our algorithm outperformed the PureSVD in all datasets. For the N=10 value in MovieLens 1M dataset, results are better than the PureSVD with a percentage of 11.70%, 9.32% in BX, 11.03% in Pinterest Image, 13.74% in R2 –

Yahoo! Music, 14.55% in Netflix and 10.23% in MovieLens 10M. The success of our algorithm in binary datasets also shows that negating the rates in **R** works well.

We also noticed a successful result in our **z-scoREC** method. Despite its simplicity, our novel method gave successful results compared to the similar ItemKNN and ICF algorithms. In Netflix and MovieLens 10M datasets, the results were better than the PureSVD. In R2 – Yahoo! Music and MovieLens 1M datasets, our method also outperformed all SVD-based methods. Considering that little N values' success is important for small screen sizes, another important aspect of our algorithm was its success in all lengths of lists.

From the *f* values in Appendix B.1a and B.1b tables, we can explore that SVD-based methods can be most suitable at low *f* values in the cold-start scenario. From here, we can deduce that the initial values in latent factors bring popular items to the forefront, and the success of cold-start algorithms depends on the number of popular products on their recommendation list. Since SVD-based applications thrive with low dimensional *f* values in cold-start user scenarios, they could be preferred in the huge recommender systems. Although our algorithm was more successful than other algorithms, it was generally close to each other and was successful at low *f* values. Especially in sparse datasets, it gave more successful results with EigenREC at close *f* values to it.

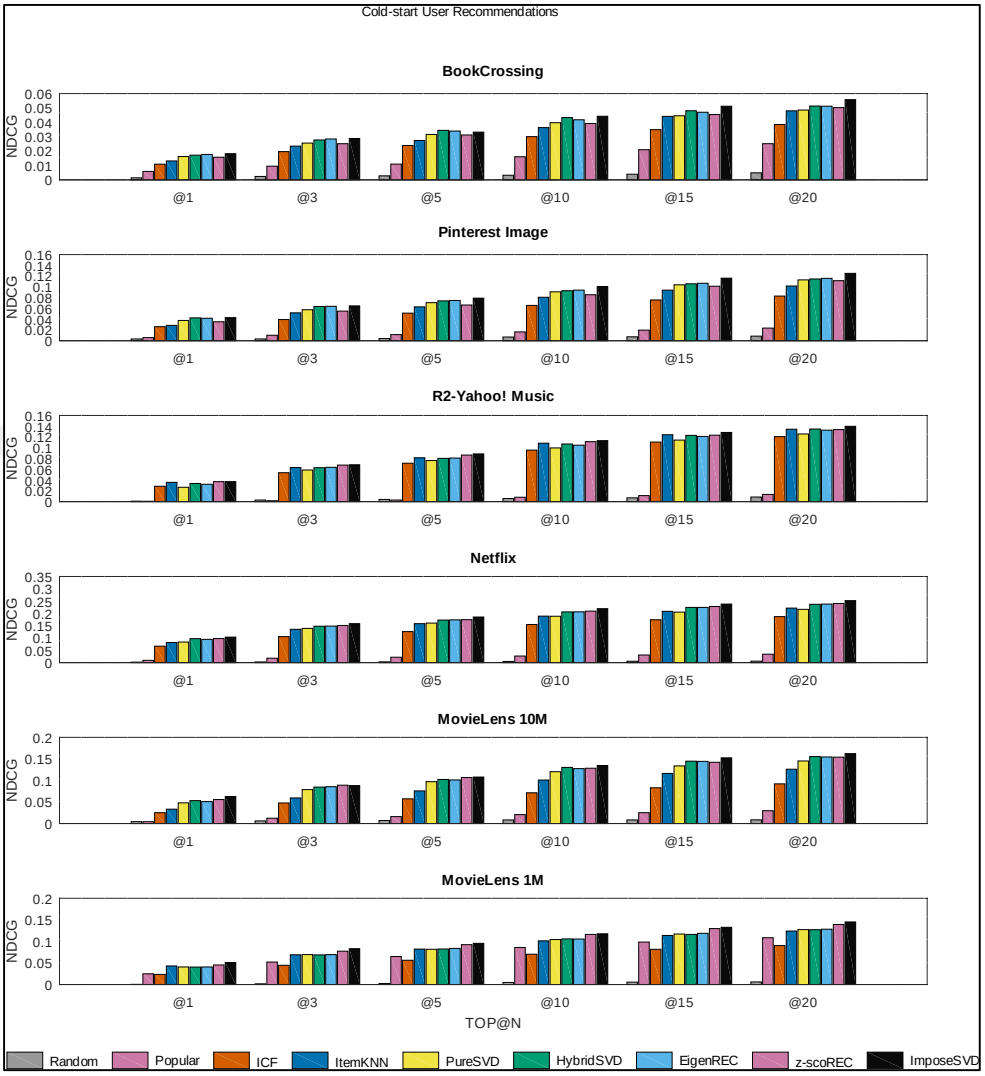

Figure 4.4 In the cold-start user scenario, the performance of the algorithms with different @N values for all datasets was evaluated with the nDCG metric

4.2.3. Cold-start system scenario

In the new systems, when users and items have inadequate ratings causing the systems have sparse data and lower density, estimating new recommendations could be hard for them. We evaluated our algorithm for new systems and created a sparsity scenario similar to denotation in EigenREC (Nikolakopoulos et.al., 2017; Nikolakopoulos et.al., 2019). We called every raw dataset the 'final stage' of a system, and randomly selected 66% of the ratings at the final stage, and called these sampled ratings as the 'previous stage'. Later, we randomly sampled 66% of ratings at the 'previous stage' and called this subset as the 'initial stage'. In our scenario 'initial stage' is the subset of the other stages and a cold-start system with lower ratings. We created the 'probe set' out of the 'initial stage' and also removed the probe set from other stages. To create 'test set' out of 'probe set', we selected the users who had rated less than 100 items and the items that were selected by at most 100 users in the 'initial stage'. We aimed to observe users and items with a few ratings and to follow their evolutions as the sparsity decreased. We evaluated all stages with the same test set.

Figure 4.5 reports the nDCG results in different sparsity for all benchmark methods for six datasets. In general, the early point where SVD-based and Item-based methods appeared to produce better results on the same test set as the sparsity value declined as expected. On the other hand, Popularity and Random algorithms continued almost at the same values at all sparsity levels and these results show that our scenario is an acceptable evaluation method. As a result, when we performed many remarks on the results via Figure 4.5, we found out that the ImposeSVD was more rewarding in all datasets and all sparsity levels than other compared algorithms. The ImposeSVD outperformed other algorithms in all results except for MovieLens 10M's and R2 – Yahoo! Music's final stages. After the rewarding results obtained in the cold-start scenario, the achievement of the z-scoREC method here was still impressive due to its purity and simplicity. We can conclude that the z-scoREC

method had better performances than the PureSVD in all experiments except for the MovieLens 10M dataset.

The most important result in Figure 4.5 is the success of the first stage values. Our algorithm outperforms all benchmark methods and the differences from other algorithms are bigger than in other stages. In the ‘initial stage’, z-scoREC is the second after the ImposeSVD in all datasets except for R2 – Yahoo! Music and Pinterest Image datasets. The results show that both our algorithms are more successful on sparse datasets.

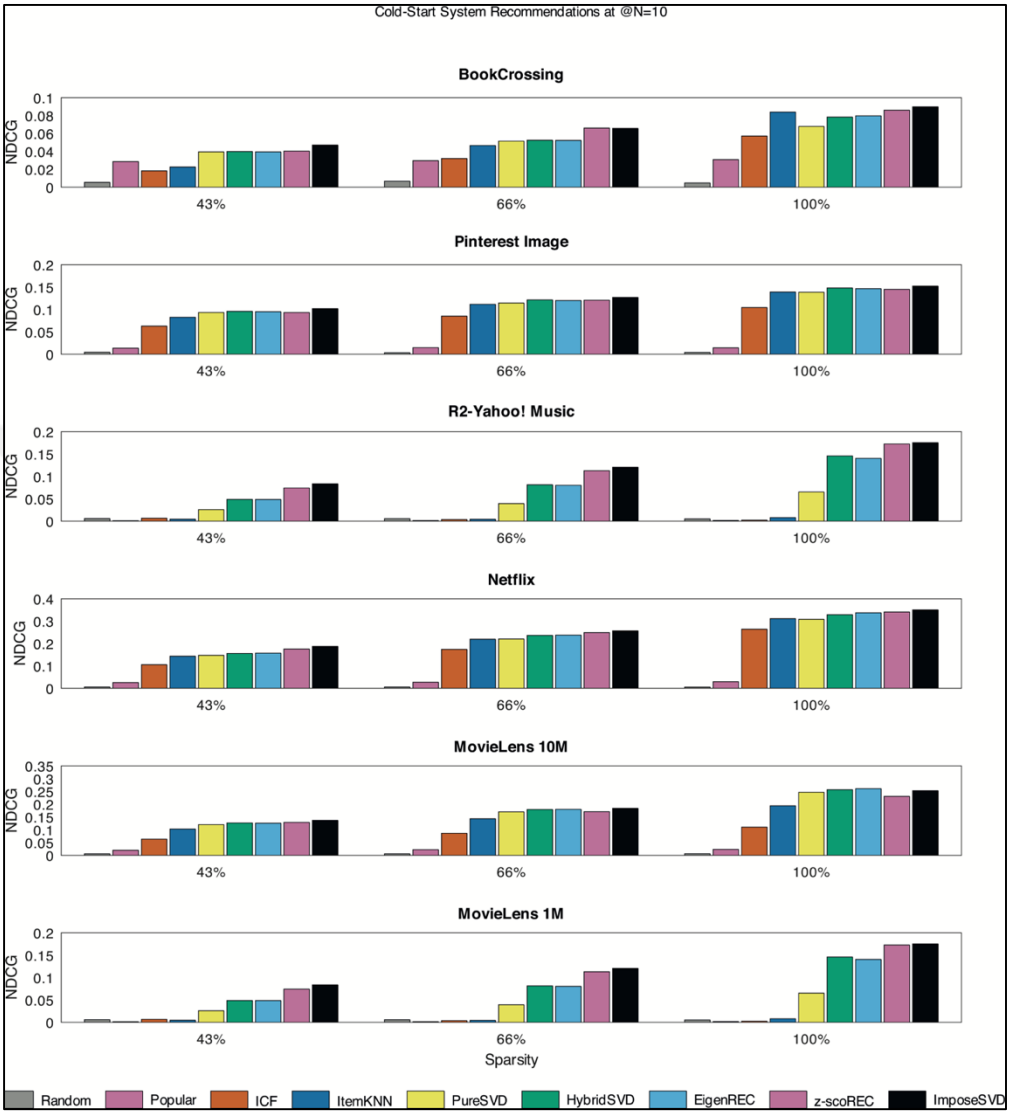

Figure 4.5 In the cold-start system's scenario, the performance of the algorithms in the @N=10 value for all datasets was evaluated with the nDCG metric

4.2.4. Long-tail items scenario

Popular items are familiar to users, and thus, recommending these items may be boring for users (Anderson, 2006). Therefore, recommending unpopular items has always been more attractive. Traditional CF algorithms dealing with relations

between popular items or popular users overshadow diverse relationships. Considering that the quality of models depends on the diversity of recommendations they offer, these CF methods might be unable to generate a diverse range of suggestions in the datasets especially when the data is inadequate (Yin et.al., 2006). The diversity can be achieved when some unpopular items are recommended to the users. Based on these arguments, we also evaluated the reaction of our algorithms for long-tail items recommendations. To obtain an experiment environment suitable for the long-tail scenario, we adopted the method in (Cremonesi et.al., 2010). As observed by the authors in the study (Cremonesi et.al., 2010), the most popular 1.7% items represent 33% of the ratings included in the Netflix dataset and they called these 1.7% items as short-head items while the remaining items are called long-tail items. Following this method (Cremonesi et.al., 2010), we sorted the items in the dataset according to their popularity in order to evaluate the existence of long-tail items in the recommendation lists. In doing so, we determined the items' popularity by their rating frequency and sorted them in descending order by the number of ratings they had. On the sorted item list according to their frequency, from top to bottom, we marked the items as 'short-head' items until the sum of the item frequencies equal to or higher than the 33% of the total ratings and marked the remaining as 'long-tail' items. In the probe set, we kept 'long-tail' items and removed the others. We created a 'test set' out of the probe set by random selection of 100 users who gave at least one full rating to 'long-tail' items. As a result, we randomly selected users with unpopular tastes for each *repeated holdout evaluation.*

According to the results presented in Figure 4.6, we observed that our algorithms outperformed the PureSVD in all datasets in a long-tail scenario. In the long-tail scenario, we aimed to evaluate the distribution of unpopular items in the recommendation lists, and as expected, the Popular baseline algorithm failed on all datasets, validating the training and test sets we created for this scenario. In the long-tail scenario, SVD-based algorithms achieved success and produced good results at high *f* values. Particularly in BookCrossing, Netflix, and MovieLens datasets, our

ImposeSVD algorithm outperformed other algorithms with a high contradict. In contrast, nearest-neighbor-based algorithms failed to find long-tail items. Our basic z-scoREC outperformed neighbor-based models with high precision and it produced nearby results to the SVD-based models.

Although long-tail item recommendation is a difficult challenge especially for SVD-based applications, as shown in (Cremonesi et.al., 2010) PureSVD can be successful at high *f* values. When we analyze the tables in Appendix B.2a and B.2b, our algorithm was successful at increasing *f* values, similar to PureSVD. In most datasets, EigenREC and HybridSVD reached their maximums at a lower *f* value with respect to the ImposeSVD, but ImposeSVD showed a significant difference in the quality of recommendation list compared to other algorithms despite the higher *f* value. Another interesting observation was that for some datasets, there is no need for shrinkage on Gram-matrix in the recommendation of unpopular items, and this process would be neglected in this scenario.

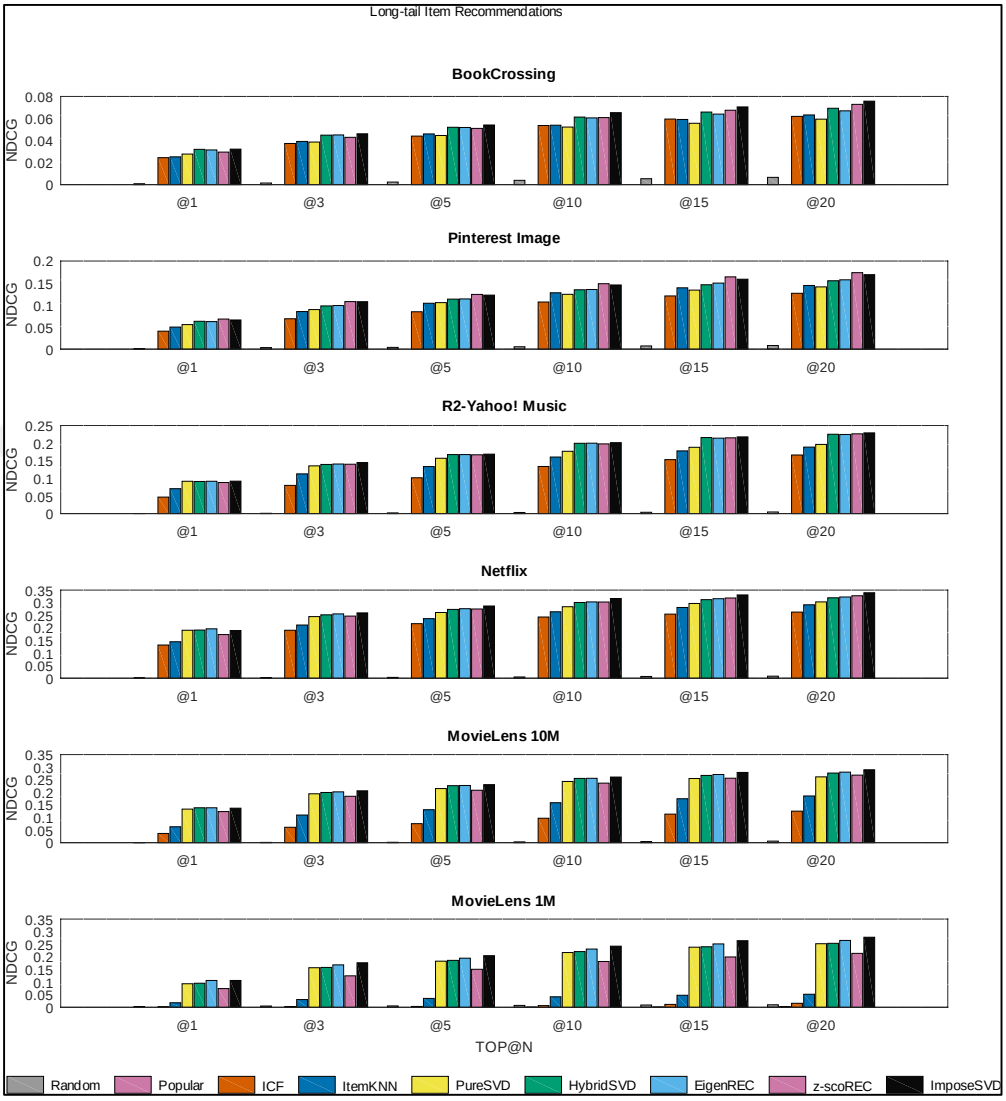

Figure 4.6 In the long-tail items scenario, the performance of the algorithms with different @N values for all datasets was evaluated with the nDCG metric

4.2.5. Effect of the lambda parameter

We evaluated our algorithms for lambda (λ) values between [-1,1] ranges with 0.05 steps in all experiments to analyze its effect in the Gram-matrix. We chose two baseline algorithms the PureSVD and the ItemKNN to compare our algorithms with because the PureSVD is the basic form of our algorithm and it has a

nonparametric form, and the ItemKNN is very similar to our z-scoREC method. We analyzed in which ranges of lambda parameters our algorithm outperformed the PureSVD and the ItemKNN. The results of the PureSVD and the ItemKNN were stable and we took the best results of each algorithm in Fig. 4.7, Fig. 4.8, and Fig. 4.9. In addition, we compared the z-scoREC algorithm with the ImposeSVD algorithm with the same parametric values to see their performance over each other.

We observed the success of our algorithms at different parameter values for the cold-start scenario in Fig. 4.7, for the long-tail item scenario in Fig. 4.8, and the cold-start systems in Fig. 4.9. Black lines represent the ImposeSVD algorithm; pink lines represent the z-scoREC algorithm. We chose the green dotted lines for the PureSVD and the blue dotted lines for the ItemKNN. The black and pink vertical dashed lines show the lambda value corresponding to the best HR value of the related algorithm. One of the noticeable observations in these results was that in the cold-start user scenario the best HR value of (λ) for each dataset differed from zero, which shows that the shifting was successful for the cold-start scenario. When the lambda value was zero, the success of both algorithms dropped dramatically. When comparing our algorithms with each other, we first observed that the ImposeSVD algorithm outperformed the z-scoREC algorithm in all parameter ranges. Our second observation was that both algorithms gave similar responses at the same parameter values. In addition, we found out that the best lambda values of our algorithms were close to each other in all data sets, and these values were especially small positive values, closer to zero in the [-1,1] ranges. In a conclusion, we inferred that the shifting process worked since the best HR points in both of our algorithms were different from zero. In the z-scoREC lines, $\lambda=0$ values showed the result of the Gram-matrix where we could see that these points were non-shifted values. Success at non-zero lambda parameters demonstrated the ability of our algorithm to capture new and strong relationships in the weight matrix. The ImposeSVD was revealed as the best algorithm for the best HR points in all datasets. The z-scoREC algorithm, on the

other hand, was more successful than the PureSVD and the ItemKNN in all datasets except for the Pinterest Image.

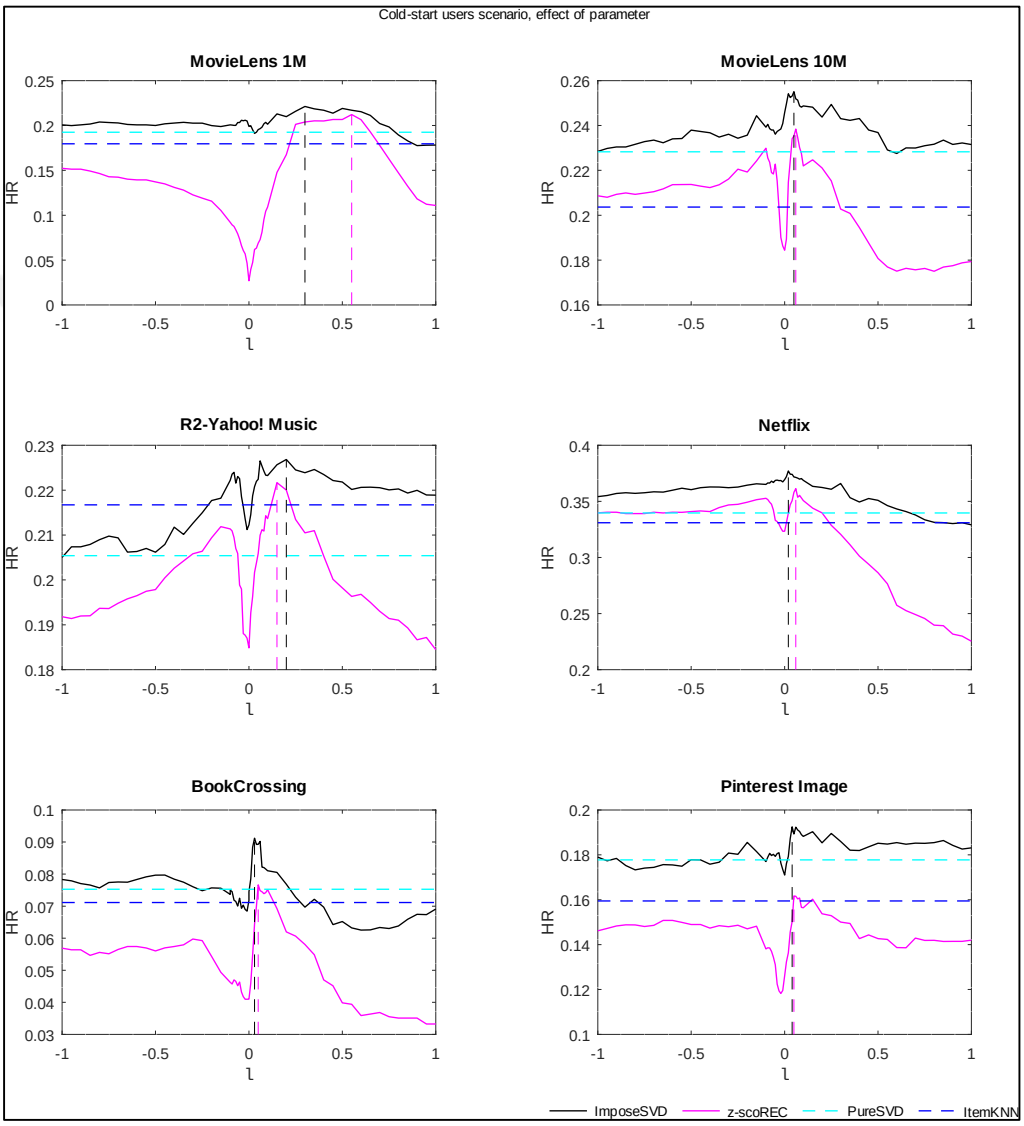

Figure 4.7. In a cold-start user scenario the effect of the lambda (λ) parameter is evaluated with the HR metric

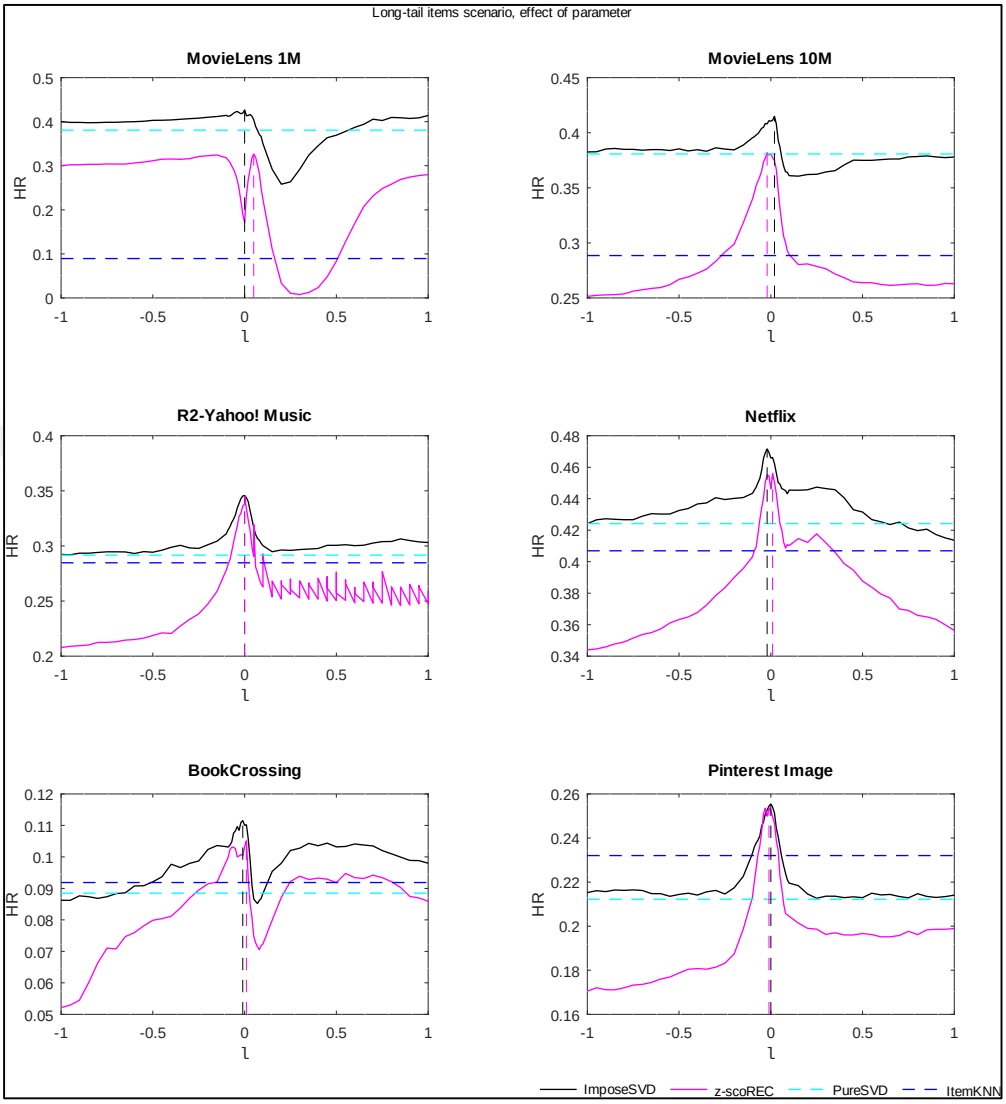

Figure 4.8. In a long-tail items scenario the effect of the lambda (λ) parameter is evaluated with the HR metric

The results of the experiments on the long-tail scenario (in Fig. 4.8) revealed that our algorithms outperformed all algorithms when λ is at zero or very close to zero in the long-tail item scenario in Fig. 4.8 while they were not successful when the λ=0 value in the cold-start user scenario as in Fig. 4.7. Therefore, we can

conclude that the shifting process had little effect on the long-tail item recommendations. However, the results indicated that the ImposeSVD was more successful in the long-tail items scenario compared to the results in the cold-start users' scenario. Although the developers of the PureSVD alleged that the PureSVD gave successful outcomes for the long-tail items at high-rank values, we found out that our ImposeSVD algorithm gave more successful results than PureSVD without shifting. In addition, the z-scoREC algorithm performed better than PureSVD in all datasets except for MovieLens datasets. In all datasets, both of our algorithms outperformed the ItemKNN algorithm. We observed successful results compared to the other algorithms at close values to zero for λ parameter in the z-scoREC algorithm. We can conclude that z-scoREC is successful without shifting in reducing the variance in the rating matrix and converging score means to the zero for item columns so that popular items reduce the pressure on other items. In this way, unknown items can come to the fore.

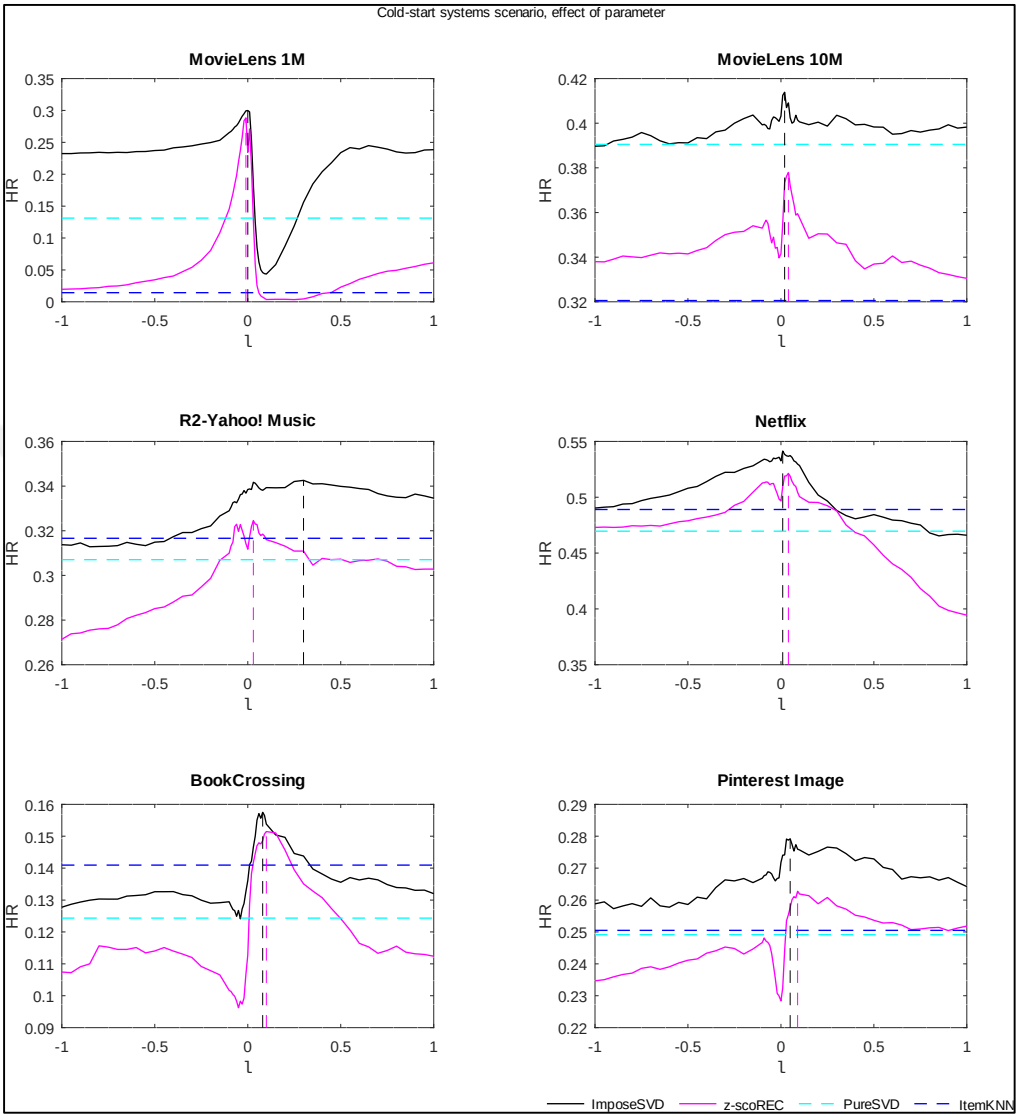

Figure 4.9. In a cold-start system scenario, the effect of the lambda (λ) parameter is evaluated with the HR metric

In the cold-start system scenario, we took the 'final stage' sampling results. This stage of the dataset performed the general recommendations for warm users who had rated less than 100 items and test items that were selected by at most 100 users. It was like a combination of cold-start user scenarios and long-tail item

recommendation scenarios. These mixed-scenario situations were a challenging task because lambda parameters behaved differently in two scenarios for our algorithms. However, in this mixed-scenario, similar to the results of the other scenarios, the ImposeSVD outperformed the PureSVD and the ItemKNN in all datasets as shown in Fig. 4.9. The z-scoREC outperformed the ItemKNN and the PureSVD in all other datasets although it was behind the PureSVD on the ML-10M dataset. Fig. 4.8 and Fig. 4.9 show similar trends for the datasets. However, lambda values were revealed to be significant because the results indicated that the best lambda values were the ones slightly greater than zero.

5. CONCLUSION

As is known, cold-start is one of the major problems in recommender systems. In the literature, much research has been done on this problem. What makes this problem important is that it has a relationship with the solution of many issues in Recommender Systems. In particular, the varying screens and richness of the interaction environments between users and products (Netflix, Spotify, Youtube, Twitch, etc.) have also demonstrated many problems identical to the cold start.

This thesis proposed to develop models for the solution of cold-start problems in various scenarios. Today's modern recommendation systems do not work only through one algorithm. Depending on the case, they could change the models or integrate different models. You may need to follow either a deterministic or heuristic method to establish new links between the user and the products. In this thesis, we have done 2 different studies, one deterministic and one heuristic, which might be used in different scenarios.

Firstly, we studied the heuristic method ACO and developed a model that gives variable recommendations to users at different times. We introduced a novel approach on how to improve item-based models with AcoRec, which is a heuristic model, and how to tune hyper-parameters with Ant Colony Optimization. By studying parameter optimization in a continuous domain, we have performed on expanding profit by exploring personalized parameters. First, we considered how to improve a model based on the specification of producing diversified and expanded recommendations to the users in their sessions, considering that common deterministic systems are no longer enough in offering recommendations for multi-line interfaces in varied domains. AcoRec remembers the heuristic approach of the ACO model, which is a successful optimization algorithm in NP-Hard problems and highlights different and neglected item recommendations.

We carried out the methods of magnifying item similarity models and how to support new ties in cases where they are inadequate. In addition, we established a model on how Ant Colony Optimization is implemented in recommendation systems

over huge graph structures in a low-dimensional manner by considering user-based micro dimensions. In this sense, parameter tuning times were minimized in our implementation, which is a big issue in ACO algorithms. To intensify these improvements, we evaluated our algorithm for the cold-start user problem and its effectiveness to recommend unpopular items.

While carrying out our experiments, we measured the quality of the recommendation lists through nDCG and R-Score, and the variety of items in the recommendation lists with the help of Coverage metrics. When we investigated the results, we noted that our study outperformed similar algorithms known in the literature in all datasets and metrics used in this study. As a result, we concluded that heuristic methods such as Ant Colony can offer rewarding results by clarifying parameter controls. We can suggest such methods can handle a variety of on-site domains.

In the second study, we focused on dimension reduction and enrichment of the rating matrix. Firstly, we introduced two novel methods, namely z-scoREC and ImposeSVD, which are more effective in cold-start user problems, cold-start system recommendations, and long-tail item recommendations than the previously proposed ICF, ItemKNN, and SVD-based methods. The results indicated that the ImposeSVD method that we presented strengthened the obvious shortcomings of PureSVD by keeping its straightforward structure and original purity in terms of processing time and computational difficulty. Evaluations displayed that the ImposeSVD model outperformed similar models on the common datasets in all experiment scenarios.

In addition to ImposeSVD methods, the other z-scoREC method we presented performed remarkably effectively as a model in item-based recommendations with cold-start users. Another significant finding of both studies was that new relationships could be discovered between items without the need for meta-information for the user and items. With the implementation of the z-scoREC method as an item-similarity in the basic item model, we brought prosperous results.

According to these analyses, the following conclusions are drawn:

1. We observed that the problem causing the cold-start issue is the sparsity of data and that the ImposeSVD and z-scoREC algorithms, which we built on enriching the data matrix with virtual new connections to reduce this sparsity, were successful in the experimental results.
2. We also observed that decreasing the data unavailability virtually is not only successful in cold-start users' scenario as well as it succeeds in recommending cold-start systems and long-tail items scenarios.
3. We tested our studies in different domains (movie, music, social media) and got flourishing results. Experiments exhibited that our models could be applied to many areas commercially.
4. In both studies, we only used implicit data and did not need demographic, personal, or track data about users, thus bypassing ethical concerns.
5. Despite its simplicity, the z-scoREC algorithm surprised us with its success. With its structure suitable for parallelization, it has shown promise that it can be successful against state-of-art research in the literature in huge data.
6. The AcoRec study has shown good results on how to add random items to recommendations and still improve the quality of the recommendation list. AcoRec showed how we can generate rich and variational recommendations and improve the quality of these recommendations when deterministic recommendation models are outdated.
7. The parameter search in a continuous distribution in the AcoRec algorithm can be used for the hyper-parameter tuning algorithms. For example, the $EASE^R$ method, which uses a single parameter, can be tested with the model we developed and can offer user-specific lambda parameters.

In our future studies, we will focus on the enrichment of z-scoREC. It may be considered to evaluate the z-scoREC algorithm with giant datasets due to its simplicity, speed, and easy implementation. Secondly, the imposed values while injecting the original data matrix could be enhanced and optimized. An algorithm that detects and eliminates unsuccessful columns in the latent factors could be investigated. In AcoRec, different item similarity matrices (Graph-based, Regression-based, $EASE^R$) could be evaluated as future works.

REFERENCES

Adomavicius, G., & Tuzhilin, A. (2005). Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. IEEE transactions on knowledge and data engineering, 17(6), 734-749.

Ahn, H. J. (2008). A new similarity measure for collaborative filtering to alleviate the new user cold-starting problem. *Information Sciences*, *178*(1), 37-51.

Anderson, C. (2006). *The long tail: Why the future of business is selling less of more*. Hachette Books.

Ar, Y., & Bostanci, E. (2016). A genetic algorithm solution to the collaborative filtering problem. Expert Systems with Applications, 61, 122-128.

Balabanović, M., & Shoham, Y. (1997). Fab: content-based, collaborative recommendation. *Communications of the ACM*, *40*(3), 66-72.

Basilico, J., & Hofmann, T. (2004, July). A joint framework for collaborative and content filtering. In *Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, pp.550-551.

Basu, C., Hirsh, H., & Cohen, W. (1998). Recommendation as classification: Using social and content-based information in recommendation. In *Aaai/iaai*, pp.714-720.

Bedi, P., & Sharma, R. (2012). Trust based recommender system using ant colony for trust computation. *Expert Systems with Applications*, *39*(1), 1183-1190.

Bellaachia, A., & Alathel, D. (2012, September). Trust-based ant recommender (T-BAR). In 2012 6th IEEE International Conference Intelligent Systems (pp. 130-135). IEEE.

Bellaachia, A., & Alathel, D. (2014, December). DT-BAR: a dynamic ANT recommender to balance the overall prediction accuracy for all users. In *CS & IT Conference Proceedings*(Vol. 4, No. 13). CS & IT Conference Proceedings.

Bellaachia, A., & Alathel, D. (2016). Improving the recommendation accuracy for cold-start users in trust-based recommender systems. International Journal of Computer and Communication Engineering, 5(3), 206.

Bennett, J., & Lanning, S. (2007). The netflix prize. In Proceedings of KDD cup and workshop 2007, 35.

Blum, C. (2005). Ant colony optimization: Introduction and recent trends. Physics of Life reviews, 2(4), 353-373.

Bobadilla, J., Ortega, F., Hernando, A., & Gutiérrez, A. (2013). Recommender systems survey. *Knowledge-based systems*, *46*, 109-132.

Bobadilla, J., Ortega, F., Hernando, A., & Gutiérrez, A. (2013). Recommender systems survey. *Knowledge-based systems*, *46*, 109-132.

Bobadilla, J., Serradilla, F., & Bernal, J. (2010). A new collaborative filtering metric that improves the behavior of recommender systems. *Knowledge-Based Systems*, *23*(6), 520-528.

Breese, J. S., Heckerman, D., & Kadie, C. (2013). Empirical analysis of predictive algorithms for collaborative filtering. *arXiv preprint arXiv:1301.7363*.

Burke, R. (2002). Hybrid recommender systems: Survey and experiments. User modeling and user-adapted interaction, 12(4), 331-370.

Candillier, L., Meyer, F., & Fessant, F. (2008). Designing specific weighted similarity measures to improve collaborative filtering systems. In *Industrial Conference on Data Mining*, pp.242-255. Springer, Berlin, Heidelberg.

Chen, Y., & de Rijke, M. (2018, October). A collective variational autoencoder for top-n recommendation with side information. In Proceedings of the 3rd workshop on deep learning for recommender systems (pp. 3-9).

Cheng, Y., Yin, L., & Yu, Y. (2014). LorSLIM: low rank sparse linear methods for top-n recommendations. In *2014 IEEE International Conference on Data Mining*, pp.90-99.

Cheng, Y., Yin, L., & Yu, Y. (2014). LorSLIM: low rank sparse linear methods for top-n recommendations. In *2014 IEEE International Conference on Data Mining*, pp.90-99.

Christakopoulou, E., & Karypis, G. (2014, May). Hoslim: Higher-order sparse linear method for top-n recommender systems. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining* pp.38-49. Springer, Cham.

Christakopoulou, E., Smith, S., Sharma, M., Richards, A., Anastasiu, D. C., & Karypis, G. (2018). Scalability and Distribution of Collaborative Recommenders.

Christoffel, F., Paudel, B., Newell, C., & Bernstein, A. (2015, September). Blockbusters and wallflowers: Accurate, diverse, and scalable recommendations with random walks. In Proceedings of the 9th ACM Conference on Recommender Systems (pp. 163-170).

Cooper, C., Lee, S. H., Radzik, T., & Siantos, Y. (2014, April). Random walks in recommender systems: exact computation and simulations. In Proceedings of the 23rd international conference on world wide web (pp. 811-816).

Cremonesi, P., Koren, Y., & Turrin, R. (2010, September). Performance of recommender algorithms on top-n recommendation tasks. In *Proceedings of the fourth ACM conference on Recommender systems,* pp.39-46.

Dacrema, M. F., Cremonesi, P., & Jannach, D. (2019, September). Are we really making much progress? A worrying analysis of recent neural recommendation approaches. In *Proceedings of the 13th ACM Conference on Recommender Systems* (pp. 101-109).

Deshpande, M., & Karypis, G. (2004). Item-based top-n recommendation algorithms. *ACM Transactions on Information Systems (TOIS)*, *22*(1), 143-177.

Dorigo, M., & Gambardella, L. M. (1997). Ant colonies for the travelling salesman problem. *biosystems*, *43*(2), 73-81.

Embarak, O. H. (2011, April). A method for solving the cold-start problem in recommendation systems. In *2011 International Conference on Innovations in Information Technology*, pp.238-243. IEEE.

Fouss, F., Saerens, M., & Shimbo, M. (2016). Algorithms and models for network data and link analysis. Cambridge University Press.

Fouss, F., Yen, L., Pirotte, A., & Saerens, M. (2006, December). An experimental investigation of graph kernels on a collaborative recommendation task. In *Sixth International Conference on Data Mining (ICDM'06),* pp.863-868. IEEE.

Frolov, E., & Oseledets, I. (2019). HybridSVD: when collaborative information is not enough. In *Proceedings of the 13th ACM Conference on Recommender Systems,*

pp.331-339.

Funk, S. (2006). Netflix update: Try this at home.

Geng, X., Zhang, H., Bian, J., & Chua, T. S. (2015). Learning image and user features for recommendation in social networks. In Proceedings of the IEEE international conference on computer vision (pp. 4274-4282).

Ghazanfar, M. A., & Prugel, A. (2013). The advantage of careful imputation sources in sparse data-environment of recommender systems: Generating improved svd-based recommendations. Informatica, 37(1).

GNU Octave, https://www.gnu.org/software/octave/about

Gohari, F. S., Haghighi, H., & Aliee, F. S. (2017). A semantic-enhanced trust based recommender system using ant colony optimization. *Applied Intelligence*, *46*(2), 328-364.

Goldberg, D., Nichols, D., Oki, B. M., & Terry, D. (1992). Using collaborative filtering to weave an information tapestry. *Communications of the ACM*, *35*(12), 61-70.

Golub, G. H., & Van Loan, C. F. (2013). Matrix computations. The Johns Hopkins University Press

Good, N., Schafer, J. B., Konstan, J. A., Borchers, A., Sarwar, B., Herlocker, J., & Riedl, J. (1999). Combining collaborative filtering with personal agents for better recommendations. *Aaai/iaai*, *439*, 1-8.

Guo, G. (2013, October). Integrating trust and similarity to ameliorate the data sparsity and cold-start for recommender systems. In *Proceedings of the 7th ACM conference on Recommender systems,* pp.451-454.

Hammar, M., Karlsson, R., & Nilsson, B. J. (2013, October). Using maximum coverage to optimize recommendation systems in e-commerce. In Proceedings of the 7th ACM conference on Recommender systems (pp. 265-272).

Harper, F. M., & Konstan, J. A. (2015). The movielens datasets: History and context. Acm transactions on interactive intelligent systems (tiis), 5(4), 1-19.

He, X., Liao, L., Zhang, H., Nie, L., Hu, X., & Chua, T. S. (2017, April). Neural collaborative filtering. In Proceedings of the 26th international conference on world wide web (pp. 173-182).

Herlocker, J. L., Konstan, J. A., Terveen, L. G., & Riedl, J. T. (2004). Evaluating collaborative filtering recommender systems. *ACM Transactions on Information Systems (TOIS)*, *22*(1), 5-53.

Hurley, N., & Zhang, M. (2011). Novelty and diversity in top-n recommendation--analysis and evaluation. *ACM Transactions on Internet Technology (TOIT)*, *10*(4), 1-30.

Jamali, M., & Ester, M. (2010). A matrix factorization technique with trust propagation for recommendation in social networks. In *Proceedings of the fourth ACM conference on Recommender systems*, pp.135-142.

Kabbur, S., Ning, X., & Karypis, G. (2013). Fism: factored item similarity models for top-n recommender systems. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp.659-667.

Kabbur, S., Ning, X., & Karypis, G. (2013). Fism: factored item similarity models for top-n recommender systems. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp.659-667.

Kaleroun, A., & Batra, S. (2014, August). Collaborating trust and item prediction with ant colony for recommendation. In 2014 Seventh International Conference on Contemporary Computing (IC3) (pp. 334-339). IEEE.

Kang, Z., Peng, C., & Cheng, Q. (2016, February). Top-n recommender system via matrix completion. In Proceedings of the AAAI Conference on Artificial Intelligence, 30(1).

Kang, Z., Peng, C., & Cheng, Q. (2016, February). Top-n recommender system via matrix completion. In Proceedings of the AAAI Conference on Artificial Intelligence, 30(1).

Karypis, G. (2001, October). Evaluation of item-based top-n recommendation algorithms. In *Proceedings of the tenth international conference on Information and knowledge management* pp.247-254.

Kim, H. N., Ji, A. T., Ha, I., & Jo, G. S. (2010). Collaborative filtering based on

collaborative tagging for enhancing the quality of recommendation. *Electronic Commerce Research and Applications*, *9*(1), 73-83.

Koren, Y. (2008, August). Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining* pp.426-434.

Koren, Y. (2009, June). Collaborative filtering with temporal dynamics. In Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining pp.447-456.

Koren, Y., Bell, R., & Volinsky, C. (2009). Matrix factorization techniques for recommender systems. *Computer*, *42*(8), 30-37.

Liang, D., Krishnan, R. G., Hoffman, M. D., & Jebara, T. (2018, April). Variational autoencoders for collaborative filtering. In Proceedings of the 2018 world wide web conference (pp. 689-698).

Liao, X., Li, X., Xu, Q., Wu, H., & Wang, Y. (2020). Improving ant collaborative filtering on sparsity via dimension reduction. Applied Sciences, 10(20), 7245.

Liao, X.; Hu, W.; Yongji, W. Ant Collaborative Filtering Addressing Sparsity and Temporal Effects. IEEE Access 2020, 8, 32783–32791.

Loh, S., Lorenzi, F., Granada, R., Lichtnow, D., Wives, L. K., & de Oliveira, J. P. M. (2009, March). Identifying Similar Users by their Scientific Publications to Reduce Cold Start in Recommender Systems. In *WEBIST*, 9, 593-600.

Malone, T. W., Grant, K. R., Turbak, F. A., Brobst, S. A., & Cohen, M. D. (1987). Intelligent information-sharing systems. *Communications of the ACM*, *30*(5), 390-402.

Massa, P., & Avesani, P. (2009). Trust metrics in recommender systems. In *Computing with social trust*, pp.259-285. Springer, London

Massa, P., & Avesani, P. (2009). Trust metrics in recommender systems. In *Computing with social trust* (pp. 259-285). Springer, London.

Nembrini, R., Ferrari Dacrema, M., & Cremonesi, P. (2021). Feature selection for recommender systems with quantum computing. Entropy, 23(8), 970.

Nikolakopoulos, A. N., & Karypis, G. (2019, January). Recwalk: Nearly uncoupled

random walks for top-n recommendation. In Proceedings of the twelfth ACM international conference on web search and data mining (pp. 150-158).

Nikolakopoulos, A. N., Berberidis, D., Karypis, G., & Giannakis, G. B. (2019, September). Personalized diffusions for top-n recommendation. In Proceedings of the 13th ACM Conference on Recommender Systems (pp. 260-268).

Nikolakopoulos, A. N., Kalantzis, V., Gallopoulos, E., & Garofalakis, J. D. (2019). EigenRec: generalizing PureSVD for effective and efficient top-N recommendations. *Knowledge and Information Systems*, *58*(1), 59-81.

Nikolakopoulos, A. N., Kalantzis, V., Gallopoulos, E., & Garofalakis, J. D. (2017, August). Factored proximity models for top-n recommendations. In 2017 IEEE international conference on big knowledge (ICBK) (pp. 80-87). IEEE

Ning, X., & Karypis, G. (2011, December). Slim: Sparse linear methods for top-n recommender systems. In *2011 IEEE 11th International Conference on Data Mining* pp.497-506. IEEE.

Ning, X., Desrosiers, C., & Karypis, G. (2015). A comprehensive survey of neighborhood-based recommendation methods. Recommender systems handbook, 37-76.

Olaleke, O., Oseledets, I., & Frolov, E. (2021, June). Dynamic modeling of user preferences for stable recommendations. In Proceedings of the 29th ACM Conference on User Modeling, Adaptation and Personalization (pp. 262-266).

Park, S. T., & Chu, W. (2009, October). Pairwise preference regression for cold-start recommendation. In *Proceedings of the third ACM conference on Recommender systems* (pp. 21-28).

Park, S. T., Pennock, D., Madani, O., Good, N., & DeCoste, D. (2006, August). Naïve filterbots for robust cold-start recommendations. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 699-705).

Parvin, H., Moradi, P., & Esmaeili, S. (2019). TCFACO: Trust-aware collaborative filtering method based on ant colony optimization. *Expert Systems with Applications*, *118*, 152-168.

Paterek, A. (2007, August). Improving regularized singular value decomposition for collaborative filtering. In *Proceedings of KDD cup and workshop*, 2007, 5-8.

Paudel, B., Christoffel, F., Newell, C., & Bernstein, A. (2016). Updatable, accurate, diverse, and scalable recommendations for interactive applications. *ACM Transactions on Interactive Intelligent Systems (TiiS)*, *7*(1), 1-34.

Pazzani, M. J. (1999). A framework for collaborative, content-based, and demographic filtering. *Artificial intelligence review*, *13*(5-6), 393-408.

Rendle, S., Freudenthaler, C., Gantner, Z., & Schmidt-Thieme, L. (2012). BPR: Bayesian personalized ranking from implicit feedback. *arXiv preprint arXiv:1205.2618*.

Resnick, P., Iacovou, N., Suchak, M., Bergstrom, P., & Riedl, J. (1994). GroupLens: an open architecture for collaborative filtering of netnews. In *Proceedings of the 1994 ACM conference on Computer supported cooperative work*, pp.175-186.

Riadi, I. C. J. (2014). Cognitive Ant colony optimization: A new framework in swarm intelligence. University of Salford (United Kingdom).

Ricci, F., Rokach, L., & Shapira, B. (2015). Recommender systems: introduction and challenges. In Recommender systems handbook (pp. 1-34). Springer, Boston, MA

Robbins, H., & Monro, S. (1951). A stochastic approximation method. *The annals of mathematical statistics*, 400-407.

S. Nadi, M.H. Saraee, M.D. Jazi, A. Bagheri Fars: fuzzy ant based recommender system for web usersIJCSI Int. J. Comput. Sci. Issues, 8 (1) (2011), pp. 203-209

Sarwar, B. M., Karypis, G., Konstan, J., & Riedl, J. (2002, December). Recommender systems for large-scale e-commerce: Scalable neighborhood formation using clustering. In Proceedings of the fifth international conference on computer and information technology (Vol. 1, pp. 291-324).

Sarwar, B. M., Konstan, J. A., Borchers, A., Herlocker, J., Miller, B., & Riedl, J. (1998, November). Using filtering agents to improve prediction quality in the grouplens research collaborative filtering system. In Proceedings of the 1998 ACM conference on Computer supported cooperative work (pp. 345-354).

Sarwar, B., Karypis, G., Konstan, J., & Riedl, J. (2000). *Application of dimensionality reduction in recommender system-a case study*. Minnesota Univ Minneapolis Dept of Computer Science.

Sarwar, B., Karypis, G., Konstan, J., & Riedl, J. (2001, April). Item-based collaborative filtering recommendation algorithms. In Proceedings of the 10th international conference on World Wide Web (pp. 285-295).

Schein, A. I., Popescul, A., Ungar, L. H., & Pennock, D. M. (2002, August). Methods and metrics for cold-start recommendations. In *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, pp.253-260.

Shambour, Q., & Lu, J. (2012). A trust-semantic fusion-based recommendation approach for e-business applications. Decision Support Systems, 54(1), 768-780.

Shani, G., & Gunawardana, A. (2011). Evaluating recommendation systems. In Recommender systems handbook, pp.257-297. Springer, Boston, MA.

Shardanand, U., & Maes, P. (1995). Social information filtering: algorithms for automating "word of mouth". In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pp.210-217.

Shenbin, I., Alekseev, A., Tutubalina, E., Malykh, V., & Nikolenko, S. I. (2020, January). Recvae: A new variational autoencoder for top-n recommendations with implicit feedback. In Proceedings of the 13th International Conference on Web Search and Data Mining (pp. 528-536).

Sobecki, J., & Tomczak, J. M. (2010, March). Student courses recommendation using ant colony optimization. In *Asian Conference on Intelligent Information and Database Systems*(pp. 124-133). Springer, Berlin, Heidelberg.

Socha, K., & Dorigo, M. (2008). Ant colony optimization for continuous domains. *European journal of operational research*, *185*(3), 1155-1173.

Son, L. H. (2016). Dealing with the new user cold-start problem in recommender systems: A comparative review. *Information Systems*, *58*, 87-104.

Steck, H. (2019, May). Embarrassingly shallow autoencoders for sparse data. In The

World Wide Web Conference, pp.3251-3257.

Stützle, T., López-Ibánez, M., Pellegrini, P., Maur, M., De Oca, M. M., Birattari, M., & Dorigo, M. (2011). Parameter adaptation in ant colony optimization. *Autonomous search*, 191-215.

Tengkiattrakul, P., Maneeroj, S., & Takasu, A. (2016, November). Applying ant-colony concepts to trust-based recommender systems. In Proceedings of the 18th International Conference on Information Integration and Web-based Applications and Services (pp. 34-41).

Tengkiattrakul, P., Maneeroj, S., & Takasu, A. (2018). Integrating the importance levels of friends into trust-based ant-colony recommender systems. International Journal of Web Information Systems.

Vahabi, H., Koutsopoulos, I., Gullo, F., & Halkidi, M. (2015, October). Difrec: A social-diffusion-aware recommender system. In Proceedings of the 24th ACM International on Conference on Information and Knowledge Management (pp. 1481-1490).

Vargas, S. (2015). Novelty and diversity enhancement and evaluation in Recommender Systems.

Weng, L. T., Xu, Y., Li, Y., & Nayak, R. (2008). Exploiting item taxonomy for solving cold-start problem in recommendation making. In *2008 20th IEEE International Conference on Tools with Artificial Intelligence*, 2, pp.113-120. IEEE.

Wu, Y., DuBois, C., Zheng, A. X., & Ester, M. (2016). Collaborative denoising auto-encoders for top-n recommender systems. In *Proceedings of the Ninth ACM International Conference on Web Search and Data Mining*, pp.153-162.

Yahoo Labs Webscope 2014. R2 - Yahoo! Music. Retrieved 07-04-2020 from https://webscope.sandbox.yahoo.com/

Yin, H., Cui, B., Li, J., Yao, J., & Chen, C. (2012). Challenging the long tail recommendation. *arXiv preprint arXiv:1205.6700*.

Zhou, Y., Wilkinson, D., Schreiber, R., & Pan, R. (2008, June). Large-scale parallel collaborative filtering for the netflix prize. In *International conference on*

algorithmic applications in management pp.337-348. Springer, Berlin, Heidelberg.

Ziegler, C. N., McNee, S. M., Konstan, J. A., & Lausen, G. (2005). Improving recommendation lists through topic diversification. In Proceedings of the 14th international conference on World Wide Web, pp.22-32.

BIOGRAPHY

Hakan YILMAZER. He received a BSc degree from Mersin University, Mersin in 2003 and an MSc degree from Mersin University, Mersin in 2013. Since 2017, he has been working as a software engineer for Çukurova University and living in Budapest/Hungary.

Publications from thesis

Yilmazer, H., & Özel, S. A. (2022). ImposeSVD: Incrementing PureSVD For Top-N Recommendations for Cold-Start Problems and Sparse Datasets. The Computer Journal.

APPENDICES

A PROOF OF LEMMA 1.

PROOF. Since $\mathbf{R}^T\mathbf{R}$ is the inner product between the same matrix $\mathbf{R}$ and also $\mathbf{R}^T\mathbf{R}$ is symmetric.

For $i=j$,

- $[\mathbf{R}^T\mathbf{R}]_{ij}$ = is a diagonal element on the intersection of the ith column and jth row.
- $[\mathbf{R}^T\mathbf{R}]_{ij}$ = number of ones in the product of ith column and jth row of $\mathbf{R}$, which is equal to the number of ones in the ith column.
- $[\mathbf{R}^T\mathbf{e}]_{ij}$ = is a diagonal element on the intersection of the ith column and jth element of $\mathbf{e}$.
- $[\mathbf{R}^T\mathbf{e}]_{ij}$ = number of ones in the product of ith column of the $\mathbf{R}$ and jth element of $\mathbf{e}$, which is equal to the number of ones in the ith column.

Therefore, the diagonal of the $\mathbf{R}^T\mathbf{R}$ matrix is the vertex degrees of the $\mathbf{R}^T$ if it's a binary matrix.

REMARK Since $\mathbf{R}^T\mathbf{R}$ is a co-citation matrix and also $\mathbf{R}\mathbf{R}^T$ is a co-author matrix, and $\mathbf{R}\mathbf{R}^T$ is symmetric too. Therefore, the diagonal of the $\mathbf{R}\mathbf{R}^T$ matrix is the vertex degrees of the $\mathbf{R}$ if it's a binary matrix hence: $\text{Diag}(\mathbf{R}\mathbf{e}^T) = \text{Diag}(\mathbf{R}\mathbf{R}^T)$

B BEST PARAMETERS AND BEST RESULTS FOR Z-SCOREC AND IMPOSESVD EVALUATIONS FOR ALL DATASETS AND SCENARIOS

In our experimental evaluation, we applied Grid-Search to find the best parameter values for each algorithm. We performed Grid-Search for each dataset separately. In the below tables, you can find the best parameter values and their associated nDCG results for each scenario when N=10.

Table B.1a Results for cold-start user scenario for MovieLens 1M, BookCrossing, and Pinterest Image datasets

	COLD-START USER SCENARIO											
	MovieLens 1M				**BookCrossing**				**Pinterest Image**			
	f/k	**α**	**d/λ**	**nDCG**	***f/k***	**α**	**d/λ**	**nDCG**	***f/k***	**α**	**d/λ**	**nDCG**
Random				0.005				0.003				0.007
Popular				0.086				0.016				0.016
ICF	20			0.070	220			0.030	10			0.066
ItemKNN	100			0.101	60			0.036	160			0.081
PureSVD	8			0.104	295			0.040	93			0.091
HybridSVD	10	0.5	0.90	0.106	175	0.1	0.75	**0.043**	135	0.1	0.85	0.093
EigenREC	10		0.20	0.106	295		0.85	0.042	123		0.55	**0.094**
z-scoREC			0.65	**0.115**			0.10	0.039			0.15	0.086
ImposeSVD	45		0.35	**0.117**	175		0.05	**0.044**	125		0.10	**0.101**

Table B.1b Results for cold-start user scenario for R2-Yahoo! Music, MovieLens 10M, and Netflix datasets

	COLD-START USER SCENARIO											
	R2-Yahoo! Music				MovieLens 10M				Netflix			
	f/k	α	d/λ	nDCG	*f/k*	α	d/λ	nDCG	*f/k*	α	d/λ	nDCG
Random				0.006				0.008				0.005
Popular				0.008				0.021				0.027
ICF	60			0.096	70			0.072	30			0.155
ItemKNN	190			0.109	200			0.101	180			0.189
PureSVD	95			0.100	54			0.120	125			0.188
HybridSVD	115	0.9	0.35	0.107	63	0.1	0.50	**0.130**	105	0.1	0.35	**0.206**
EigenREC	115		0.40	0.105	63		0.50	0.128	130		0.30	**0.206**
z-scoREC			0.15	**0.111**			0.10	0.127			0.05	0.205
ImposeSVD	155		0.20	**0.113**	105		0.05	**0.133**	140		0.05	**0.217**

Table B.2a Results for long-tail items scenario for MovieLens 1M, BookCrossing, and Pinterest Image datasets

	LONG-TAIL ITEMS SCENARIO											
	MovieLens 1M				BookCrossing				Pinterest Image			
	f/k	α	d/λ	nDCG	*f/k*	α	d/λ	nDCG	*f/k*	α	d/λ	nDCG
Random				0.007				0.004				0.005
Popular				0.000				0.000				0.000
ICF	10			0.006	10			0.054	10			0.107
ItemKNN	20			0.041	90			0.054	60			0.128
PureSVD	200			0.217	575			0.052	625			0.124
HybridSVD	180	0.1	0.95	0.221	300	0.1	0.05	**0.061**	250	0.99	0.00	0.135
EigenREC	225		1.30	**0.231**	225		-0.25	**0.061**	350		0.00	0.135
z-scoREC			0.05	0.175			0.00	**0.061**			0.00	**0.149**
ImposeSVD	220		0.00	**0.243**	725		0.00	**0.065**	700		0.00	**0.150**

Table B.2b Results for long-tail items scenario for R2-Yahoo! Music, MovieLens 10M, and Netflix datasets

	LONG-TAIL ITEMS SCENARIO											
	R2-Yahoo! Music				**MovieLens 10M**				**Netflix**			
	f/k	**α**	**d/λ**	**nDCG**	***f/k***	**α**	**d/λ**	**nDCG**	***f/k***	**α**	**d/λ**	**nDCG**
Random				0.003				0.004				0.005
Popular				0.000				0.000				0.000
ICF	10			0.134	10			0.097	10			0.243
ItemKNN	90			0.161	50			0.159	210			0.264
PureSVD	1200			0.177	500			0.243	825			0.284
HybridSVD	275	0.1	0.00	**0.200**	300	0.1	0.35	**0.256**	400	0.1	0.50	0.301
EigenREC	275		0.00	**0.200**	300		0.35	**0.256**	425		0.15	**0.303**
z-scoREC			0.00	0.195			0.05	0.236			0.05	0.299
ImposeSVD	1050		0.00	**0.202**	650		-0.05	**0.259**	1350		-0.05	**0.313**

Table B.3a Results for sparsity final stage scenario for MovieLens 1M, BookCrossing, and Pinterest Image datasets

	SPARSITY %100											
	MovieLens 1M				**BookCrossing**				**Pinterest Image**			
	f/k	**α**	**d/λ**	**nDCG**	***f/k***	**α**	**d/λ**	**nDCG**	***f/k***	**α**	**d/λ**	**nDCG**
Random				0.005				0.005				0.004
Popular				0.002				0.031				0.014
ICF	10			0.002	10			0.057	10			0.104
ItemKNN	20			0.008	170			0.084	230			0.139
PureSVD	600			0.065	40			0.068	275			0.139
HybridSVD	90	0.5	0.05	0.146	100	0.1	0.55	0.078	70	0.99	0.45	**0.148**
EigenREC	1050		1.35	0.141	125		0.45	0.080	70		0.35	0.147
z-scoREC			-0.01	**0.171**			0.20	**0.086**			0.06	0.145
ImposeSVD	1050		-0.01	**0.174**	150		0.05	**0.090**	150		0.03	**0.152**

Table B.3b Results for sparsity final stage scenario for R2-Yahoo! Music, MovieLens 10M, and Netflix datasets

	SPARSITY %100											
	R2-Yahoo! Music				**MovieLens 10M**				**Netflix**			
	f/k	**α**	**d/λ**	**nDCG**	***f/k***	**α**	**d/λ**	**nDCG**	***f/k***	**α**	**d/λ**	**nDCG**
Random				0.005				0.006				0.006
Popular				0.009				0.023				0.029
ICF	10			0.119	10			0.111	10			0.264
ItemKNN	190			0.178	250			0.194	300			0.311
PureSVD	375			0.174	125			0.247	275			0.308
HybridSVD	400	0.1	0.25	**0.193**	200	0.1	0.55	**0.257**	225	0.1	0.40	0.329
EigenREC	400		0.20	**0.194**	225		0.35	**0.261**	525		0.15	**0.338**
z-scoREC			0.03	0.184			0.04	0.229			0.02	0.337
ImposeSVD	375		0.06	**0.193**	175		0.05	0.252	675		0.04	**0.347**

Table B.4a Results for sparsity second stage scenario for MovieLens 1M, BookCrossing, and Pinterest Image datasets

	SPARSITY %66											
	MovieLens 1M				**BookCrossing**				**Pinterest Image**			
	f/k	**α**	**d/λ**	**nDCG**	***f/k***	**α**	**d/λ**	**nDCG**	***f/k***	**α**	**d/λ**	**nDCG**
Random				0.006				0.007				0.003
Popular				0.001				0.030				0.015
ICF	10			0.004	10			0.032	10			0.085
ItemKNN	40			0.004	70			0.047	200			0.112
PureSVD	950			0.039	70			0.052	125			0.115
HybridSVD	200	0.5	0.00	0.081	10	0.1	0.85	0.053	90	0.1	0.55	**0.122**
EigenREC	900		0.95	0.078	60		0.65	0.052	90		0.55	0.120
z-scoREC			-0.01	**0.111**			0.08	**0.066**			0.07	0.121
ImposeSVD	950		0.00	**0.119**	60		0.07	**0.066**	150		0.20	**0.127**

Table B.4b Results for sparsity second stage scenario for R2-Yahoo! Music, MovieLens 10M, and Netflix datasets

	SPARSITY %66											
	R2-Yahoo! Music				**MovieLens 10M**				**Netflix**			
	f/k	**α**	**d/λ**	**nDCG**	***f/k***	**α**	**d/λ**	**nDCG**	***f/k***	**α**	**d/λ**	**nDCG**
Random				0.005				0.006				0.006
Popular				0.009				0.022				0.027
ICF	10			0.104	10			0.086	10			0.174
ItemKNN	90			0.140	200			0.143	180			0.219
PureSVD	525			0.124	125			0.170	225			0.220
HybridSVD	275	0.1	0.55	0.139	125	0.1	0.60	0.179	225	0.5	0.55	0.236
EigenREC	275		0.45	0.140	125		0.50	**0.180**	225		0.55	0.238
z-scoREC			0.05	**0.144**			0.04	0.171			0.06	**0.245**
ImposeSVD	650		0.03	**0.148**	175		0.03	**0.184**	350		0.04	**0.253**

Table B.5a Results for sparsity initial stage scenario for MovieLens 1M, BookCrossing, and Pinterest Image datasets

	SPARSITY %33											
	MovieLens 1M				**BookCrossing**				**Pinterest Image**			
	f/k	**α**	**d/λ**	**nDCG**	***f/k***	**α**	**d/λ**	**nDCG**	***f/k***	**α**	**d/λ**	**nDCG**
Random				0.006				0.005				0.004
Popular				0.001				0.029				0.014
ICF	10			0.007	10			0.018	30			0.063
ItemKNN	10			0.005	220			0.023	250			0.083
PureSVD	1100			0.026	125			0.039	100			0.093
HybridSVD	800	0.99	0.60	0.049	10	0.1	0.75	**0.040**	100	0.1	0.90	**0.096**
EigenREC	1150		0.95	0.047	10		0.75	0.039	100		0.90	0.095
z-scoREC			-0.01	**0.073**			0.08	**0.040**			0.09	0.093
ImposeSVD	1200		-0.01	**0.082**	75		0.09	**0.047**	90		0.15	**0.102**

Table B.5b Results for sparsity initial stage scenario for R2-Yahoo! Music, MovieLens 10M, and Netflix datasets

	SPARSITY %33											
	R2-Yahoo! Music				**MovieLens 10M**				**Netflix**			
	f/k	**α**	**d/λ**	**nDCG**	***f/k***	**α**	**d/λ**	**nDCG**	***f/k***	**α**	**d/λ**	**nDCG**
Random				0.005				0.006				0.006
Popular				0.009				0.020				0.025
ICF	10			0.073	10			0.064	10			0.105
ItemKNN	180			0.095	190			0.103	130			0.143
PureSVD	100			0.086	60			0.120	80			0.147
HybridSVD	125	0.1	0.40	**0.103**	40	0.99	0.70	0.127	175	0.1	0.55	0.156
EigenREC	125		0.25	0.102	70		0.45	0.126	175		0.55	0.157
z-scoREC			0.09	0.102			0.06	**0.129**			0.05	**0.170**
ImposeSVD	225		0.02	**0.111**	150		0.15	**0.137**	225		0.05	**0.183**

C. ACO BASED RECOMMENDER SYSTEMS

Table C.1. Studies About Ant-Colony Based Recommender System

Research	**Trust based recommender system using ant colony for trust computation. (2012)**
Abbreviation	TARS
Score	Ranking Based; TOP_N@(1,2,5)
Metrics	Recall, Precision, F1-Score
Method	Merged Trust Based User Confidences and Pearson as trust values between users. In ACO sense, nodes denoted users and edges are trust values. Paper finds best neighbors for a user in ACO, and predict new rating scores from these neighbors using Resnick's (1994) rating prediction formula
Scenario	Cold-start, Sparsity
Benchmarks	UCF (Pearson, k=100) (Resnick, 1994)
Datasets	Jester, MovieLens 100K
Evaluation	Precision is increased by approximately 8% and 3% respectively using TARS at time t=0 and approximately by 12% and 8%
References	(Bedi and Sharma, 2012)

Table C.2. Studies About Ant-Colony Based Recommender System

Research	**A semantic-enhanced trust-based recommender system using ant colony optimization. (2017)**
Abbreviation	STARS
Score	Ranking Based; TOP_N@(10)
Metrics	MAE (for Parameter Tuning), Recall, ARHR
Method	Semantically Clustering Items, Trust Based User Similarity Merge (MSD-Jaccard, Pearson), In ACO sense, nodes denoted users and edges are trust values. Paper finds best neighbors for a user in ACO, and predict new rating scores from these neighbors using Resnick's (1994) rating prediction formula.
Scenario	Cold-start, General Recommends, Sparsity, MMIC problem
Benchmarks	UCF (Resnick,1994), ICF (Sarwar et.al., 2001), TARS (Bedi and Sharma, 2012), TSF (Shambour & Lu, 2012)
Datasets	MovieLens 100K (ML-100K), MovieLens 1M (ML-1M)
Evaluation	Against to TARS

Dataset	Scenario	Recall	ARHR
ML-100K	General	9.33%	14.63%
ML-1M	General	14.24%	12.10%
ML-100K)	Cold-start User	19.30%	18.66%
ML-1M	Cold-start User	47.08%	55.43%
ML-100K	MMIC	15.01%	18.98%
ML-1M	MMIC	13.18%	14.89%

References	(Gohari et.al., 2017)

Table C.3. Studies About Ant-Colony Based Recommender System

Research	**TCFACO: Trust-aware collaborative filtering method based on ant colony optimization (2019)**
Abbreviation	TCFACO
Score	Rating Prediction
Metrics	MAE, RMSE, Coverage
Method	Trust Based UCF merged with user sim(Tau* and Pearson)
Scenario	Cold-start, Heavy users, Opinion users, Long-tail items, Contra(mean(ratings)<1.5), General
Benchmarks	ItemAverages, UserAverages, TARS, TrustSVD, TrustMF, SocialMF, SVD++
Datasets	FilmTrust, Epinions, Ciao
Evaluation	Against to TARS Dataset / Scenario / MAE Epinions / Cold-start user / 0.837/0.853 Ciao / Cold-start user / 0.723/0.701
References	(Parvin et.al., 2019)

Table C.4 Studies About Ant-Colony Based Recommender System

Research	**Trust-based ant recommender (T-BAR) (2012)**
Abbreviation	T-BAR
Score	Rating Prediction
Metrics	MAE, Coverage
Method	Trust Based UCF
Scenario	Cold-start, Heavy users
Benchmarks	UCF (Pearson) (Resnick, 1994), MT (Mole Trust) (Massa & Avesani, 2009)
Datasets	Epinions
Evaluation	Against to Mole Trust Dataset / MAE / Coverage T-Bar / 1.459 / 93.% Mole Trust / 0.673 / 11% DT-Bar / 0.714 / 55%
References	(Bellaachia & Alathel, 2012)

Table C.5 Studies About Ant-Colony Based Recommender System

Research	**DT-BAR: a dynamic ANT recommender to balance the overall prediction accuracy for all users (2014)**
Abbreviation	DT-BAR
Score	Rating Prediction
Metrics	MAE, Coverage
Method	Trust-Based UCF
Scenario	Cold-start, Heavy users
Benchmarks	T-BAR (Bellaachia & Alathel, 2012), MT (Mole Trust) (Massa & Avesani, 2009)
Datasets	Epinions
Evaluation	Against to Mole Trust Dataset / MAE / Coverage T-BAR / 1.459 / 93.% Mole Trust / 0.673 / 11% DT-BAR / 0.714 / 55%
References	(Bellaachia & Alathel, 2014)

Table C.6 Studies About Ant-Colony Based Recommender System

Research	**Student courses recommendation using ant colony optimization.**
Score	Rating Prediction
Metrics	MAE, NMSE and other error predictions
Method	Probability based similarity
Scenario	General
Benchmarks	UCF (Pearson) (Resnick, 1994), User-Based CBF
Datasets	Real data from the University of Information System EdukacjaCL
References	(Sobecki & Tomczak, 2010)

Table C.7 Studies About Ant-Colony Based Recommender System

Research	**Applying ant-colony concepts to trust-based recommender systems. (2016)**
Score	Rating Prediction
Metrics	MAE, Coverage
Method	User Similarity (User factors from SVD-U and trustworthiness of among them)
Scenario	General
Benchmarks	ALT-BAR
Datasets	Epinions
Evaluation	Coverage percentage for ALT-BAR %28.8 and for their method %70.8
References	(Tengkiattrakul et.al., 2016)

Table C.8 Studies About Ant-Colony Based Recommender System

Research	**Integrating the importance levels of friends into trust-based ant-colony recommender systems. (2018)**
Score	Rating Prediction
Metrics	MAE, Coverage
Method	User Similarity (User factors from SVD-U and trustworthiness of among them)
Scenario	High Accuracy, Coverage
Benchmarks	ALT-BAR
Datasets	Epinions
Evaluation	Coverage for ALT-BAR, %28.8, for their method %70.8
References	(Tengkiattrakul et.al., 2018)

Table C.9 Studies About Ant-Colony Based Recommender System

Research	**Improving the recommendation accuracy for cold-start users in trust-based recommender systems. (2016)**
Abbreviation	ALT-BAR
Score	Rating Prediction
Metrics	MAE, Coverage
Method	Averaged Localized Trust-Based Ant Recommender
Scenario	Cold-start, Heavy-user
Benchmarks	UCF (Pearson) (Resnick, 1994), MT (Mole Trust) (Massa & Avesani, 2009), T-BAR (Bellaachia & Alathel, 2012)
Datasets	Epinions
Evaluation	Cold Start MAE; ALT-BAR: 0.502, T-BAR:1.459, MT:0.674, CF:1.094 Coverage; ALT-BAR: 56%, T-BAR: 97%, MT: 18%, CF: 3%
References	(Bellaachia & Alathel, 2016)

Table C.10 Studies About Ant-Colony Based Recommender System

Research	**Collaborating trust and item-prediction with ant colony for recommendation (2014)**
Score	TOP_N@(10)
Metrics	Recall, Precision, F1-Score
Method	Similar to TARS, only item deviation distance products in prediction formula.
Scenario	General, Shilling Attack, Cold-start Users, Sparse Matrix, Grey Sheep Users
Benchmarks	UCF (Pearson) (Resnick, 1994)
Datasets	MovieLens 100K
Evaluation	General scenario using Recall metric; UCF: 32.11% Original Paper: 33.14%
References	(Kaleroun & Batra, 2014)

Table C.11 Studies About Ant-Colony Based Recommender System

Research	**Ant Collaborative Filtering Addressing Sparsity and Temporal Effects (2020)**
Abbreviation	ACF
Score	Rating Based and Ranking Based
Metrics	Rating Based; RMSE, Evaluation Time Ranking Based; Precision, Recall, Accuracy
Method	Compute User Pheromones, Item Pheromones, Then Compute Predictions for new Rating and Rankings
Scenario	Sparsity, Over Specification
Benchmarks	Rating Based; UCF, ICF, NMF, PLSA, RSM Ranking Based; UCF, ICF, NBI, RSM, BM25-Item
Datasets	Douban, LastFM
Evaluation	Precision Metric Results Dataset | Douban | LastFM UCF | 0.045 | 0.045 ICF | 0.006 | 0.040 ACF | 0.062 | 0.076
References	(Liao et.al., 2020a)

Table C.12 Studies About Ant-Colony Based Recommender System

Research	**Improving ant collaborative filtering on sparsity via dimension reduction. (2020)**
Abbreviation	IACF
Score	Rating Based and Ranking Based
Metrics	Rating Based; RMSE, Evaluation Time Ranking Based; Precision, Recall, Accuracy
Method	Upgrade to ACF, Adding Clusters to ACF
Scenario	Sparsity
Benchmarks	Rating Based; UCF, ICF, NMF, PLSA, RSM, ACF Ranking Based; UCF, ICF, NBI, RSM, BM25-Item, ACF
Datasets	ML 10M, ML 1M, Douban, NetEase
Evaluation	**Precision Metric Results** **Dataset** | Douban | NetEase **ACF** | 0.057 | 0.076 **IACF** | 0.070 | 0.081
References	(Liao et.al., 2020b)

Table C.13 Studies About Ant-Colony Based Recommender System

Research	**Fars: fuzzy ant-based recommender system for web users. (2011)**
Abbreviation	FARS
Score	Rating Based
Metrics	Precision, Recall
Method	Jaccard Similarity between users and Fuzzifying user-item matrix
Scenario	URL Recommendation
Benchmarks	ACO (Ant Based)
Datasets	Web Logs
Evaluation	Recall ACO 0.030 Fars 0.033
References	(Nadi et.al., 2011)