KADİR HAS UNIVERSITY
SCHOOL OF GRADUATE STUDIES
PROGRAM OF COMPUTER ENGINEERING

IMPROVING THE QUALITY OF RECOMMENDER SYSTEMS IN E-COMMERCE PLATFORMS

MUHİTTİN IŞIK
PROF. DR. HASAN DAĞ

PHD DISSERTATION

İSTANBUL, JUNE, 2021

IMPROVING THE QUALITY OF RECOMMENDER SYSTEMS IN E-COMMERCE PLATFORMS

MUHİTTİN IŞIK
PROF. DR. HASAN DAĞ

PHD DISSERTATION

A THESIS SUBMITTED TO THE GRADUATE STUDIES WITH THE AIM TO MEET THE PARTIAL REQUIREMENTS REQUIORED TO RECEIVE A PhD IN THE DEPARTMENT OF ENGINEERING AND NATURAL SCIENCE

İSTANBUL, JUNE, 2021

NOTICE ON RESEARCH ETHICS AND
PUBLISHING METHODS

I, MUHİTTİN IŞIK;

- hereby acknowledge, agree and undertake that this PhD Dissertation that I have prepared is entirely my own work and I have declared the citations from other studies in the bibliography in accordance with the rules;
- that this PhD Dissertation does not contain any material from any research submitted or accepted to obtain a degree or diploma at another educational institution;
- and that I commit and undertake to follow the "Kadir Has University Academic Codes of Conduct" prepared in accordance with the "Higher Education Council Codes of Conduct".

In addition, I acknowledge that any claim of irregularity that may arise in relation to this work will result in a disciplinary action in accordance with the university legislation.

ACCEPTANCE AND APPROVAL

This study, titled **IMPROVING THE QUALITY OF RECOMMENDER SYSTEMS IN E-COMMERCE PLATFORMS**, prepared by the **MUHİTTİN IŞIK**, was deemed successful with the **UNANIMOUS** as a result of the thesis defense examination held on the **03.06.2021** and approved as a **PHD THESIS** by our jury.

JURY: SIGNATURE:

Prof. Dr. Hasan DAĞ (Advisor) (Kadir Has University) _____

Prof. Dr. Songül VARLI (Co-Advisor) (Yıldız Technical University) _____

Assoc. Prof. Dr. Tamer DAĞ (Co-Advisor) (Kadir Has University) _____

Assoc. Prof. Dr. Taner ARSAN (Kadir Has University) _____

Prof. Dr. Mustafa BAĞRIYANIK (İstanbul Technical University)_ _____

I confirm that the signatures above belong to the aforementioned faculty members.

_____

(Title, Name and Surname)

Director of the School of Graduate Studies

APPROVAL DATE: …/…/…

TABLE of CONTENTS

IMPROVING THE QUALITY OF RECOMMENDER SYSTEMS
IN E-COMMERCE PLATFORMS

ABSTRACT

Especially Covid-19 pandemic process, which has taken the world by storm, has shed a clear light on the place of e-commerce, which is already increasing its influence with the globalizing world, in the future world of commerce. This pandemic process has shown that companies that can carry out their business on the internet, regardless of their sector, may survive, and the rest may suffer a great deal. As people choose the way to meet even their daily needs online, it caused the majority of companies to quickly turn to and analyze this field and start to take their place in the e-commerce world as soon as possible. As such, the tricks of e-commerce systems gained great importance. The most prominent of these tricks are product ratings and reviews that completely change the shopping idea of users. The scope of this research consists of experimental studies on how to calculate these product rating systems, which change the profit margin of the companies in the world of e-commerce, more effectively. Our study suggests different methods for the calculation of product rating score to prevent fake accounts, biased or malicious users and companies that make guiding or deceptive interventions for their products and services. That is to say, our study includes alternatives to the primitive calculations used in such systems that can be used in different e-commerce platforms, which can perform various calculations based on reliable users, time factor and reviews of products. In our experimental studies, we have reached various results that can prevent both the deceptive and guiding effect of fake accounts and the primitive and inadequacy of calculation methods in e-commerce systems. In this way, it is aimed that individuals who shop in various e-commerce platforms can reach real information and values and help companies that offer products and services to mirror themselves and create intelligent business ideas.

Keywords: E-commerce, Recommender Systems, Trust Rank, Time Decay, Rating Score, Trustful Users, Review Rating, Quality Star Rating

E-TİCARET SİSTEMLERİNDE
TAVSİYE SİSTEMLERİNİN KALİTESİNİ ARTTIRMAK

ÖZET

Tüm dünyayı kasıp kavuran Covid-19 pandemisi, globalleşen dünya ile birlikte zaten etkisini her geçen gün artıran e-ticaretin, gelecek ticaret dünyasındaki yerine net bir ışık tutmuştur. Bu pandemi süreci, hangi sektörde olunursa olunsun işlerini bir nebze de olsa internet üzerinden yürütebilen şirketlerin ayakta kalabileceğini, geriye kalanların ise büyük bir hezimete uğrayabileceklerini gösterdi. Günümüzde insanlar günlük ihtiyaçlarını bile internet üzerinden giderme yolunu seçince, şirketlerin büyük bir kısmının bu alana yönelmesine, bu alanı çözümlemesine ve bir an önce e-ticaret dünyasındaki yerini almaya çalışmasına yol açmıştır. Hal böyle olunca e-ticaret sistemlerinin püf noktaları büyük bir önem kazandı. Bu püf noktalarının en göze çarpanları ise kullanıcıların alışveriş fikrini tamamen değiştiren ürün puanları ve yorumları olmuştur. Bu araştırmamızın kapsamı da e-ticaret dünyasında bulunan şirketlerin kâr marjını değiştiren bu ürün puanlama sistemlerinin nasıl daha sağlıklı hesaplanacağına dair deneysel çalışmalardan oluşmaktadır. Çalışmamız ürün puanlama sistemlerindeki kötü ve ön yargılı kullanıcıları, sahte hesapları ve kendi ürün ve servislerine yönelik yaptıkları yönlendirici veya aldatıcı müdahalelerde bulunan şirketleri önlemek amacıyla birbirinden farklı yöntemler önermektedir. Bir başka deyişle, çalışmamız güvenilir kullanıcılar, zaman faktörü ve ürünlerin yorumlarından yola çıkarak çeşitli hesaplamalar yapabilen, farklı e-ticaret ortamlarında ürün puanlama sistemleri için kullanılan temel hesaplamalara alternatif yöntemler içermektedir. Yaptığımız deneysel çalışmalarda e-ticaret ortamlarındaki ürün puanlama sistemlerinde yaşanan gerek sahte hesapların aldatıcı ve yönlendirici etkisini gerekse hesaplama yöntemlerinin ilkelliğini ve yetersizliğini önleyecek çeşitli sonuçlara ulaşılmıştır. Böylece hem çeşitli e-ticaret ortamlarında alışveriş yapan bireylerin gerçek bilgi ve değerlere ulaşması hem de ürün ve hizmet sunan şirketlerin kendilerine ayna tutmasına ve yeni iş fikirleri oluşturmasına yardımcı olunması hedeflenmiştir.

Anahtar Sözcükler: E-ticaret Sistemleri, Tavsiye Sistemleri, Güven Oranı, Oylama Puanı, Güvenilir Kullanıcılar, Yorum Puanlama, Yıldız Puanlama

ACKNOWLEDGEMENT

I would like to express my deep and sincere gratitude to my advisor Prof. Hasan DAĞ for his guidance, encouragement, patience, and continuous support throughout the work of this research. I would not have been able to do the research and achieve learning in the same manner without his help and support. His recommendations and instructions have enabled me to assemble and finish the dissertation effectively.

Anneme…

LIST OF TABLES

LIST of FIGURES

1. INTRODUCTION

With the increasing time spent by the people in the electronic environment with each passing day, the trade has changed its direction. E-commerce, which we can roughly describe as the online version of traditional trade, increases its market share day by day with many alternatives it offers. Whether it is due to the limited time of today's people or because it contains more options and convenience, it makes e-commerce more attractive and preferred. These advantages have been applied not only to users or customers but also to companies that provide products or services. Thanks to e-commerce, small businesses have had the opportunity to survive by reaching more masses in a short period of time against dominated conglomerates that it has not been able to fight with traditional ways. The biggest advantages of e-commerce are that it can be accessed by users 24/7, requires less labor and expense costs for companies compared to traditional commerce, brings a wide variety of products, less costly than traditional shopping, transparent business systems, personalized customer experiences, accessible from all over the world, which are the signals that trade will shift in this direction over time.

With the increasing market share of e-commerce, it has led companies to invest more in this area, whether to reach more audiences or retain existing customers. For this purpose, companies have tried many ways such as adding their website to search engines' databases, advertising on social media, providing quality customer service, creating simple, understandable, authentic and reliable content, and the most important one is to make new, interesting, user-specific, and reliable recommendations which are the main subject that we work on in this thesis. Let's explain roughly recommendation systems to understand the concept of the thesis.

Recommendation systems have become an indispensable building block for companies to be successful in e-commerce platforms because they have the advantage to offering the right product at the right time to the right person by using them. Recommendation engines are roughly software tools for providing next best offer, next best decision or next best activity suggestion for a particular customer. These suggestions, decisions or offers help customers or users to make a decision in numerous fields such as when

choosing a music to listen, when buying an item, when selecting a movie for watching or when trying to find a book similar with one before. Amazon, Netflix, eHarmony, Pandora, TripAdvisor are probably the most well-known examples that use recommendation systems [1] .

There are a couple of techniques in recommendation systems such as non-personalized recommenders, content-based recommenders, collaborative filtering recommenders etc. In this thesis, we focus on content-based recommenders and collaborative filtering recommenders which are used especially in e-commerce systems.

Content-Based Filtering: *"Content-based recommender systems base recommendations on user ratings and similarity between items"* [2]. Actually, this approach comes from information retrieval. Namely, it is based on content analysis. This content can be a document or a website, or it can be defined as a movie, music, or a restaurant. It tries to provide items that are similar to those that users preferred before. In order to recommend new items, this algorithm compares attributes of items by looking for a user profile in which preferences are pre-existed in the database. In other words, it is actually based on the prosperities of the products and a profile of the customer's personal preferences or interests.

This algorithm usually uses the Term Frequency & Inverse Document Frequency (tf-idf) weighting in order to summarize the features of an item in databases. The tf-idf value reflects how important a word in a document or in other words how many times a word appears in a document. It is often used by search engines to calculate how much a document is related to a given query [3].

Collaborative Filtering: Collaborative Filtering is a technique in recommendation systems, especially used by the biggest websites like Amazon, Netflix, Pandora and others, that uses user behavior such as purchases, clicks, and ratings. In this way, it provides recommendations to users using user items such as movies, music, books, etc. Collaborative filtering has a couple of algorithms to provide recommendations. In this thesis, we focus on especially user-based and item-based algorithms. These two ways of generating recommendations are typical.

1.1 Problem Definition

As the number of people shopping on the internet increases, the number of investments and researches in this area has also increased. In particular, the companies try to develop new strategies to increase their sales by examining the behavior of the people using their sites. For instance, according to the report [4], 95% of the respondents stated that they consulted customer reviews before buying any products. They also stated that product reviews are very important especially in products such as electronics (82%), appliances (80%), and computers (80%) that have high prices. Again, in the same research, 80% of users stated that they examined especially negative reviews while buying any products. According to the research results of [5], when purchasing any product, the product score and the number of the people scoring to the related product are taken into consideration by the users. Especially if a product has above 4.5 stars and reviewed by many people, the rate of buying that product increases even more. Interestingly, if a product has received a low rating, it is not preferred by users regardless of the number of reviews, even if the product is rated by only two people. According to the results of another study in [6], 98% of the users stated that they checked the reviews when they thought about buying a product and 60% of them checked often or quite often. Many types of research and reports that examine the behaviors of users on e-commerce platforms while purchasing a product or service have been examined in certain parts of our thesis.

When researches and reports have shown that users pay close attention to product ratings and reviews when purchasing a product or service, this has prompted both companies and users to manipulate the results of recommender algorithms. Some companies started to create fake accounts on their respective e-commerce platforms and give high scores to their products or services, especially this is a problem for many recommender systems which have databases that suffer from sparsity problem. Likewise, biased bloggers or malicious users have tried similar ways to lower the ratings of companies they dislike. Apart from this, there are also errors arising from traditional recommender systems. Since some recommender algorithms prefer traditional methods when calculating product score, it is realized that products or services do not get the values they deserve. The reason for this may be some challenges such as cold start, sparsity, scalability, overspecialization, robustness problems encountered by recommender systems as well as logical errors . For example, as we

explain in the related Chapter 4 of our thesis, the values of some products or services may change over time. For instance, when calculating the star rating of a hotel, it may be a problem to keep a rating that was given ten years ago, and a rating given yesterday. Although the hotel has renovated itself completely over time, it may seem to be a hotel that offers bad services since it could not get rid of the stars given years ago and because of that it may not get the value it deserves. The opposite is also true. A very bad hotel due to the high stars it collected years ago can get high scores, which will affect the users most.

In order to be a solution to the problems mentioned above in the recommendation systems, we suggest three different models that can be used in different e-commerce platforms in calculating the product or service rating score.

The first one is to find reliable or trusted users based on the relationship between users and calculate the product rating score considering the trust values of users. In other words, as the trust value of a user who rated the related product increases, the effect of that user on the product rating score increases.

The second one is based on time decay. As we mentioned above, the structure or value of some products may change over time. Perhaps the value of a movie or a piece of music may not change over time, but the quality of the companies offering services such as hotels, restaurants, may change over time. Namely, the red-hot ratings and reviews made to such businesses enable us to access the healthiest information about their products or services. In this model, which is considered for such e-commerce platforms, the score of the product changes according to the date of the user rating. In other words, as the user's rating date gets closer to the present day, its effect on the product rating score also increases.

Our third model is about the reviews made on the products carefully examined by the users as we mentioned, that is on sentiment analysis. We find the rating score of the products based on the reviews made on the products. After investigating the effect of many texts preprocessing to accomplish this, a hybrid model is proposed in addition to already existing models such as review-based, sentence-based, and dictionary-based.

1.2 Aim and Objectives

The main goal of this thesis is to research alternative methods to improve the quality of traditional product star ratings used in recommender systems in e-commerce platforms. In order to achieve this, there are three distinctive objectives.

The first objective is to overcome the negative effects of fake accounts on e-commerce platforms. As we know that because of the sparsity problems in databases, fake accounts can easily affect the results of recommender algorithms especially a product that does not have enough votes by users. This situation represents a great sense for e-commerce platforms especially when considering that majority of companies have less than 1% density of databases [7]. For this purpose, it is proposed a recommender model which finds the users who are trustful and have a great effect on other's opinion by analyzing the relationship between users. With the proposed model, the recommender systems are expected to provide recommendations to customers based on trustful users' opinions to improve the quality of the recommender system in e-commerce platforms. In this way, customers may be less regret if the product doesn't have good quality as expected when they buy or consume a product. Besides that, some products don't have enough ratings in order to calculate their real rating score. Sometimes products get very high or very low rating scores. Therefore, we can get the real rating score of a product by looking at trustful users' opinions.

The second objective is to calculate the product rating score based on time decay of users' ratings which is another proposed alternative method in place of the traditional calculation of the product rating score. As we know that the quality of some products or services changes over time and hot ratings give us more reliable information about the related products or services. For this reason, when the rating score of any product is calculated, the time decay of the ratings of each user who rated to product is considered. In this way, the effects of the old ratings given to the relevant product are broken.

The third objective, which is another proposed alternative method in place of the traditional calculation to overcome the negative effects of the fake accounts and to improve the quality of product rating score, is to calculate the product rating score based on sentiment reviews analysis. For this purpose, the reviews are first processed in a series of preprocessing methods with explaining each preprocessing method's effect on the calculation of the rating score. Then, besides traditional sentiment analysis methods,

the scores of the products are tried to be calculated with the proposed hybrid model and comparisons are made between the related methods such as review-based, sentence-based, and dictionary-based sentiment analysis. Besides, the classifiers that best adapt to the sentiment analysis methods used are tried to be determined.

1.3 Contributions

The main contributions of the thesis are as follows:

In Chapter 3, the rating scores of the products are tried to be calculated by finding reliable or called trustee users in the system. Thus, it is tried to show how both fake accounts affect e-commerce platforms and how to overcome these types of problems with the relevant method. For this purpose, it is analyzed the relationship between users, and it is calculated a trustworthiness value for each of them. In this way, the rating score of each item is calculated by a weighted average of users' ratings according to their trustworthiness values instead of getting a direct average of the users' ratings. According to the results, there is a great difference rating score between average rating score and weighted rating score based on trustee users on the items, which are rated by between 2-20 users. On the other hand, when the number of users increases, especially more than 100 people, the difference between the average rating score and weighted rating score based on trustee users decreases almost "0". It means that when the number of users who rated the item decreases, the effect of the fake accounts goes up. Consequently, if we think databases that are suffering especially from the sparsity problems, this model can be a nice solution. By this model, items may get a deserved rating scores more than in the traditional models.

In Chapter 4, besides the trust-based calculation, to improve the quality of product rating score and reducing the effects of old ratings, the rating scores of the products in the dataset are tried to be calculated by finding time decay of users' ratings when creating the list of recommendations by the recommender. After introducing the concept time decay of users' ratings by explaining its mathematical definition, the product rating scores are calculated, and the effects of time decay of the users' ratings are analyzed via experimental results. In this way, we try to overcome some weaknesses of the traditional rating score calculation since ratings of the users given to the related products can be too old or vice versa. This is important because some products or

services may change over time and hot ratings can be more important when calculating the rating score of those products. Besides this, both time-based and trust-based methods are executed together to analyze the results. According to the results, if a product is rated by trustee users and their ratings' date is up to date enough, this indicates that the product received the score it deserves.

Chapter 5 focuses on the impact of simple text preprocessing decisions in order to predict fine-grained review rating stars. The aim of this chapter is to analyze preprocessing techniques and their influence, at the same time explain the interesting observations and results on the performance of a five class-based review rating classifier. According to the experimental results, a simple stopwords elimination, lowercasing, removing common words, and lastly the combination of 1-to-3 Ngrams perform better than other preprocessing methods for improving the classification accuracy of the five class-based review rating stars on the restaurant reviews. Besides this, results show that the effects of the preprocessing methods can change in any domain. For this reason, all the possible preprocessing methods should be considered to apply before used in any application. And applying the order of the preprocessing methods can also be important. For instance, the effects of abbreviations, acronyms, stemming and lemmatization might be higher after executing lowercasing to the related text.

In Chapter 6, it is tried to overcome the quality of multi-class star rating challenges, specifically on restaurant reviews, to calculate the overall star ratings via sentence-based, review-based, dictionary-based, and proposed hybrid-based sentiment analysis methods. It is observed that the Support Vector Machines classifier gives better results compared to other classifiers in determining the star ratings of the restaurants based on the text of the reviews. In fact, the point to be considered is which classifier performs better according to the sentiment analysis approach chosen. Otherwise, according to the experimental results, the Logistic Regression and Multinomial Naïve Bayes also performed closely, without selecting the sentiment analysis approach. When we come to sentiment analysis approaches, we see that the Dictionary Based method gives worse results compared to other methods, except that it gives the best result in the Decision Tree classifier. The Sentence Based method seems good in terms of the number of reviews correctly it classifies, but it is not successful in distributing this number equally

to each class. The Review Based method gives the best results in classifiers such as Random Forest and Logistic Regression compared to other methods. In addition, the number of correct predictions and the almost equal division of this number into each class shows that this method is successful than the Dictionary Based and Sentence Based methods. Similarly, Hybrid Based method gives the best results in Multinomial Naïve Bayes and Support Vector Machines classifiers compared to other methods. With its result in the Support Vector Machines classifier, it gives the best result among all other classifiers. It also gives the best result in the average error rate of all classifiers. Apart from this, it gives the best result in the number of correct predictions and the success of distributing this number almost equally to each class.

1.4 Thesis Outline

Chapter 1 presents the possible solutions and roughly the results we obtained via experimental studies by shedding light on the reasons and needs in the emergency of the current thesis.

Chapter 2 describes the building blocks, the most used methods, and the working principle of the Collaborative Filtering recommender system, which is one of the most used methods in recommendation systems. It also examines the most used similarity metrics in this field and the most common challenges.

Chapter 3 describes an alternative method that can be used to calculate product rating score in recommendation systems. The mathematical background of the method, which we define as trust-based calculation, is explained first, and then the experimental results are shared by clarifying how this method is applied to the recommendation systems.

Chapter 4 describes another alternative method that can be used to calculate product rating score in recommendation systems. First of all, the studies in this field are examined and then the mathematical background of the method, which we define as the time-based method, is explained. Then, the implementation of this method to the recommendation systems is explained, and also the experimental results are shared by combining with the trust-based method.

Chapter 5 deals with product reviews which we can use in sentiment analysis to calculate product rating scores. However, before making the calculations, it explains the text pre-processing methods and experimental results are shared showing how the text

pre-processing methods affect the success of the classifiers used to calculate product rating score.

Based on the previous section, Chapter 6 examines the existing methods of sentiment analysis and presents a hybrid model based on these methods in calculating the product rating score. In addition, the classifiers used in applying with these methods are examined and their success in calculating the product rating score is compared.

2. COLLABORATIVE FILTERING RECOMMENDER SYSTEM

Collaborative Filtering (CF) recommender systems are based on the logic that similar groups exhibit similar behaviors as in real life. Mankind may exhibit far more exceptional situations rather than other living groups due to human complex behaviors, but this does not impair general perception. People tend to be divided into different groups in many fields ranging from the football team held among themselves, the style of music listened to, the type of movie watched, the style of dress worn, etc. In other words, people most likely show the same behavior as the group they are in. Based on this logic, CF recommender systems try to find similar users and similar products from the data set obtained from any platform to advise users. For example, when trying to find similar users, it analyzes users' behaviors, such as clicks, ratings, or purchases starting from the products chosen by similar groups of users. As a result, CF recommender systems perform a series of analysis operations based on users' past behavior, calculate similarities between users and their behavior, and advise users in the next step. Before explaining the methods used in CF recommender systems, Let's explain the general working structure.

CF consists of some basic components as shown in Fig. 2.1. You can see this structure, especially in Mahout's Taste library. The Taste library has a fast and flexible structure and offers many options for user-based and item-based recommender systems. These basic structures that makeup CF can be briefly described as follows:

DataModel: The memory created for preferences, products, and users.

User-Similarity: The interface that defines the similarity between users.

Item-Similarity: The interface that defines the similarity between products.

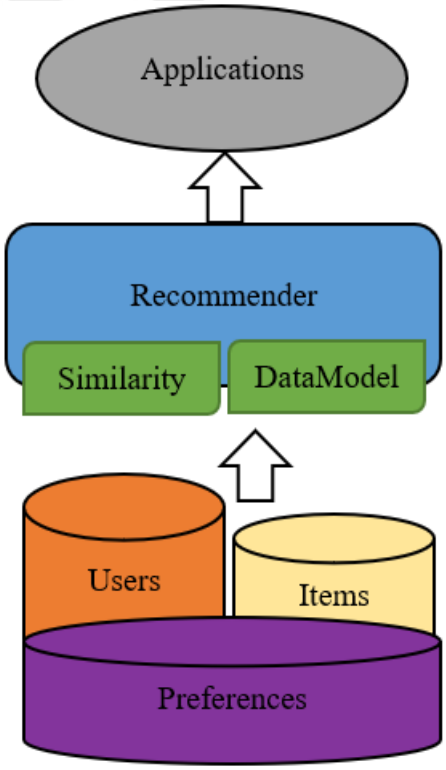


Figure 2.1 The Structure of Collaborative Filtering

Recommender: The interface for providing recommendations.

UserNeighborhood: The interface that enables neighboring relationships of similar users to be used by the recommender.

The CF recommender has the opportunity to compare items and users with the help of the UserNeighborhood interface obtained through the DataModel. It uses these similarity results to offer target users as recommendations. These recommendations can be a numerical value called prediction like predicting a score of a movie in a movie dataset or can be a list of top N movies called recommendation that the target user may like to watch. As it is seen from the Fig. 2.1, CF recommender uses three kinds of objects: users, items, and preferences which are the relationship between users and items.

Items can be any objects that are recommended such as books, movies, songs, computers, cars, mobile phones, etc., or services such as hotels, restaurants, supermarkets, transportation. Therefore, the item data may include different features/attributes that identify these products or services like genre, movie title, director, actors, year released, IMDB score of a movie.

Users are those who evaluate, criticize, rate the items, or express their feelings about the products or services. Therefore, the user data may contain a lot of personal information that identifies users such as age, gender address, educational status, socioeconomic status, occupation, etc.

Preferences express users' opinions about relevant products or services. The recommender engine can explicitly or implicitly obtain users' ideas about items. If the recommender engine tries to understand one's opinion through actions on the platform without prompting the user, then the implicit method is used. For example, starting from the products that the user navigates through the relevant platform, it perceives what kind of products the user likes, and then it offers similar products to that user. However, in the explicit method, the recommender engine asks the user directly to express his opinion/feelings about the item on a certain scale. The type of the used scale (called a rating) may vary like:

Binary ratings: The system asks users to give a negative or positive opinion about an item, whether it is good or bad, like or dislike.

Ordinal ratings: The system asks users to comment on an item with rated expressions such as very bad, bad, neutral, good, very good, excellent.

Numerical ratings: The system prompts users to rate an item with a numeric rating. For example, rating a movie out of 10 on a movie platform like Netflix, or rating hotels from 1 to 5 stars on a travel agency platform like TripAdvisor.

Knowing how these three different types of objects that the recommender system uses are related to each other and how they look like on the data set can help us to understand how the CF recommender works.

Our data set has a structure as in Table 2.1. The location of the rows and columns can be changed, and multiple categories can be included. As it is realized, columns represent users, while rows represent products. To give an example, the second user rates the first item with the value 2. Similarly, the first user does not rate the item 1 but rates the second item with the value 4. By the way, each individual rating is within a numerical scale, 1 means the user doesn't like the relevant item while 5 means the user likes a lot, and 0/- means that the user has not yet rated the relevant item. In our study, we use a similar data set, based on the ratings given by users, similarities are found, item recommendations are presented, and lastly, results are examined.

Table 2.1 User – Item Rating Matrix

Users

Items	U_1	U_2	U_3	U_4	U_5	…	U_n
I_1	-	2	-	3	5	…	3
I_2	4	-	5	-	3	…	5
I_3	5	-	3	-	4	…	-
I_4	-	2	5	-	-	…	-
I_5	3	1	-	5	-	…	2
I_6	5	1	4	-	5	…	2
…	…	…	…	…	…	…	…
I_m	-	2	3	-	5	…	mn

To understand the basic logic of the CF recommender, we can give a small example from the Table 2.1. For example, when we look at the second item carefully, we see that the users U_1, U_3, U_5 rate the item I_2. Likewise, we see the same users, U_1, U_3, U_5, rate the items I_3, I_6. The CF recommender makes meaning out of these three products "If a

group of similar users rates three different items together, they are likely to be similar. The same logic can be considered for the opposite of this situation. If different items are rated by two or more similar users, there is a high likelihood of similarity between those items. CF recommender applies these types of similarity techniques between items and users in order to build a recommender system. Now let's take a look at the raw state of the data set that we use in our study before creating the item-user matrix.

Table 2.2 The data set structure used in the research

User_id	Item_id	Ratings
1	1	3
1	2	4
1	3	4
1	4	5
…	…	…
$User_n$	$User_m$	$Ratingr_{nm}$

Table 2.2 shows the score given by each user in the dataset to items that they rated for. For example, the first user rate the first item in the dataset with 3 points out of 5. Likewise, the first user rate the second item in the dataset with value 4. The CF recommender makes use of this table to create the user-item rating matrix and it calculates similarities between users or items, then gives item recommendations to users.

Collaborative filtering recommender systems can be further subdivided into two main categories which are model-based and memory-based approaches. The model-based CF approach uses data mining techniques such as dimensionality reduction, regression, and clustering to make item recommendations. That is, this approach calculates the relationship between items through the user-item rating matrix and creates a model to estimate the user's scores for the items. Since model-based CF recommender uses data mining methods, this approach has also been a solution to the sparsity problems associated with recommender algorithms. The memory-based approach is also called neighborhood-based CF. As the name implies, it makes calculations by looking at the relationships between users or items and then makes recommendations to the users based on these calculations. This approach uses mostly two techniques: User-based and Item-based CF. In this section, we give some details about these two techniques.

2.1 User-based Collaborative Filtering

The basic logic of the user-based CF is to search out the other users that are most similar (nearest-neighbors) to a target user and drawing conclusions from their experiences based on their close proximity in order to recommend some items in any fields such as games, movies, books, songs, etc. *Namely, "a subset of users is chosen based on their similarity to the active user, and a weighted combination of their ratings is used to produce predictions for this user"* [8]. User-based CF is the most common technique in recommender systems and the general operating principle of the User-based CF is as shown in Fig. 2.2 [9];

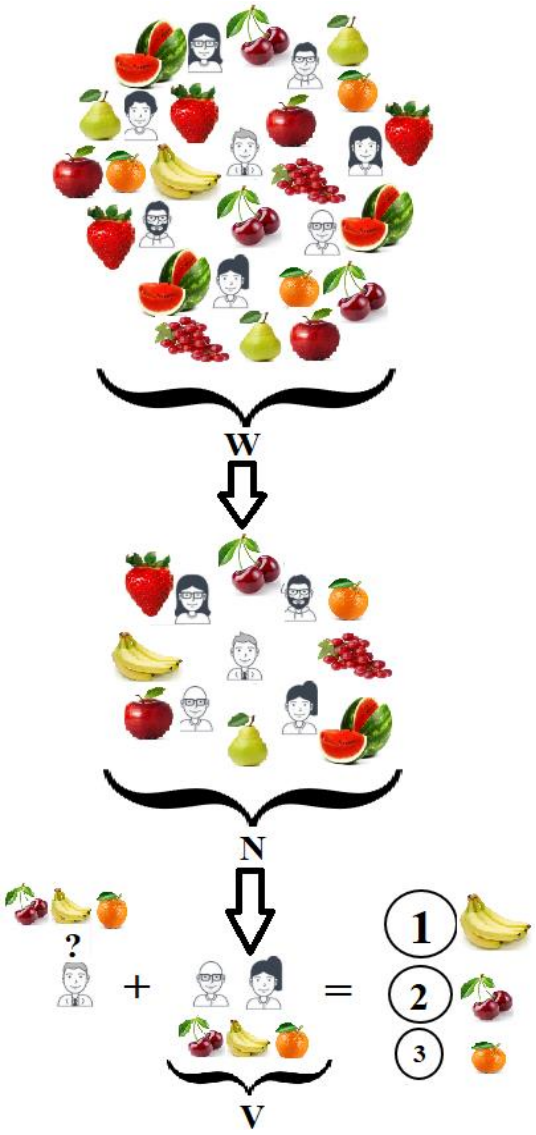


Figure 2.2 Visual representation of the user-based CF

Algorithm 2.1:

For every other user W
    Compute a similarity s between u and W
    Retain the top users, ranked by similarity, as a neighborhood N

For every item i that some user in N has a preference for,
    But that u has no preference for yet
    For every other user V in N that has a preference for i
        Compute a similarity s between u and V
        Incorporate V's preference for i, weighted by s, into a running average

As you can see from Algorithm 2.1, there are three nested loops. In the first loop, we calculate the similarities (s) between our target user (u) and all other users (symbolized as "W") in our dataset. We then select the users (symbolized as "N") that are the most similar to our target user in the amount we previously determined (this amount can be a certain number of users, or it can be the sum of users exceeding a certain similarity threshold.). In the second loop, we select the users who experience the products we intend to recommend to the target user among the most similar users we have selected ("N"), and we eliminate the rest. In the last loop, we have users (symbolized as "V") who are familiar with the product we intend to offer, and they are similar to our target user. We calculate the weighted score of the products, and we recommend based on the similarity rates of the nearest neighbors ("V"). Assuming similarities between users have been calculated, let us take an example with Formula (2,1) given below.

$$r_{ui} = \frac{\sum_{v \epsilon V_i(u)} W_{uv} r_{vi}}{\sum_{v \epsilon V_i(u)} |W_{uv}|} \tag{2.1}$$

In the Formula (2,1), user-based CF tries to estimate the rating (r_{ui}) of the target user (u) for the new item (i). Accordingly, user-based CF multiplies the score (r_{vi}) given to the relevant item by the similar users (nearest-neighbor) with the similarity rate (W_{uv}) between our target user and similar users and then divides the result by the sum of the similarity rates. Let's clarify this with a small example.

Table 2.3 Scores given by the users to the movies

	Gladiator	**Godzilla**	**Troy**	**King Kong**	**Braveheart**
Sheldon	4	1	5	2	?
Maria	2	5	2	4	1
Rena	3	2	4	4	-
Uygar	5	2	4	1	4

Our question is whether or not user-based CF should recommend to Sheldon to watch the movie “Braveheart”, which he has not seen before. Let’s take a look at Table 2.3 before finding the answer to this question with the Formula 2.1. As it is seen, the user Sheldon and the user Uygar both watched the movies in Table 2.3 and had similar tastes. Both users gave high ratings for the movies “Gladiator” and “Troy”, while low ratings for the movies “Godzilla” and “King Kong”. According to this observation, if Uygar has rated the movie “Braveheart” with value “4”, Sheldon will probably love the movie and give it a high score. We can also think of the opposite. Sheldon and the user Maria, for example, gave completely different scores to the same movies. Sheldon rated the movie “Godzilla” with value “1”, Maria rated with value “5”, or Sheldon rated the movie “Troy” with the value “5”, while Maria rated with value “2”. This situation shows us that Sheldon and Maria are two opposing characters about movie tastes. Based on this situation, if Maria rated with the value “1” to the movie “Braveheart”, Sheldon would probably give a high score. Let’s calculate with the given Formula 2.1.
Suppose that the similarities between Sheldon and neighbors are already calculated (how the similarity can be calculated are explained in Section 2.3) and they are 0,85 with Uygar, 0,15 with Maria respectively.

$$r = \frac{0,85 * 4 + 0,15 * 1}{0,85 + 0,15} = 3,55$$

As a result, Sheldon would likely give a high score to this movie. In fact, as we explained before, we try to find the most similar users and then apply this formula. In this example, we had to take user Maria, who is not very similar to our target user, because we have a limited number of users. Therefore, the result is below than the score given by user Uygar. If we had calculated over much more similar users, this value could have risen.

2.2 Item-based Collaborative Filtering

The basic logic of this technique is that if a product similar to the products that a user has previously chosen is recommended, she/he will most probably like it. Namely, it’s like recommending a new horror movie to someone who likes to watch horror movies. Item-based CF recommender and the content-based recommender shouldn’t be

confused with each other. In the content-based approach, the contents of the products are similar, namely attributes/features of the products that are similar. But the content of the products may not be similar in the item-based CF approach. For instance, those who buy smart television, they also buy a TV unit. Whereas one product is in the digital field, the other is in the field of furniture. Like the story you may have heard before in market basket analysis, it is found that an unexpected correlation between the sales of diapers and beer in the same transaction. As a result, item-based CF calculates the similarity between items instead of users in order to make recommendations. Thus, instead of finding similar users when giving a recommendation to a user, it tries to find out similar items by using his/her likes. The general operating principle of the item-based CF is as follows [9];

Algorithm 2.2:

For every item i that u has no preference for yet
    For every item j that u has a preference for
        Compute a similarity s between i and j
        Add u's preference for j, weighted by s, to a running average
Retain top items, ranked by weighted average

As shown in Algorithm 2.2, in the first loop, we first identify the products (i) that our target (u) user has not experienced before. Then, in the second loop, we calculate the similarities (s) between the products our target user experiences (j) and not experiences (i) before. Then we offer the products that are ordered according to the weighted average calculated by the similarities to the target user respectively. To illustrate this, suppose that the similarities between items have been calculated, let us take an example with the given Formula 2.2 below.

$$r_{ui} = \frac{\sum_{j \in V_u(i)} W_{ij} r_{uj}}{\sum_{j \in V_u(i)} |W_{ij}|} \tag{2.2}$$

In Formula (2.1), user-based CF tries to estimate the rating (r_{ui}) of the target user (u) for the new item (i). Accordingly, in Formula (2.2), item-based CF multiplies the score (r_{uj}) given to the relevant item by the target user with the similarity rate (W_{ij}) between items and then divides the result by the sum of the similarity rates. Let's clarify this with a small example.

Table 2.4 Scores given by the target user to the movies

	Gladiator	Godzilla	Troy	King Kong	Braveheart	Kingdom of Heaven
Sheldon	4	1	5	2	?	?

As it is seen from Table 2.4, Sheldon has not watched both the movie "Braveheart" and the movie "Kingdom of Heaven". The answer we're trying to find is which movie should be the first to recommend to our target user. In other words, which one of these movies we should recommend first that our target user is more likely to click on it to watch. This time we use the item-based CF technique. As we mentioned earlier, we first find the weighted average of each movie based on the similarities between the products and order them according to the results. For this purpose, suppose that the similarities between the movies to be recommended and the other movies in our dataset are already calculated (how the similarity can be calculated is explained in Section 2.3) and are shown below in Table 2.5.

Table 2.5 Similarities between movies

	Gladiator	Godzilla	Troy	King Kong	Braveheart	Kingdom of Heaven
Gladiator	1	0,2	0,8	0,1	0,9	0,8
Godzilla		1	0,1	0,9	0,2	0,2
Troy			1	0,2	0,7	0,9
King Kong				1	0,2	0,1
Braveheart					1	0,7
Kingdom of Heaven						1

Let's predict the ratings with given Formula (2.2).

$$r_{Braveheart} = \frac{0,9*4+0,2*1+0,7*5+0,2*2}{0,9+0,2+0,7+0,2} = 3,85$$

$$r_{KingdomofHeaven} = \frac{0,8*4+0,2*1+0,9*5+0,1*2}{0,8+0,2+0,9+0,1} = 4,05$$

As a result, Sheldon would likely give a high score to both movies. But according to the results, it seems that the movie "Kingdom of Heaven" is more preferable. In fact, as we

explained before, we try to find the most similar items and then apply the Formula (2.2). In this example, we had to take the movies “Godzilla” and “King Kong”, which are not very similar to our target movies, because we have a limited number of items. If we had calculated over much more similar movies, we could have achieved more satisfactory results.

2.3 Exploring Similarity Metrics

The most important part of the recommender algorithms is the similarity implementations. Both content-based and collaborative filtering (user-based and item-based) use several types of similarity metrics. It is very important to determine the right similarity metric according to your data otherwise these approaches may fail. Since these components are quite important, we explain their basics briefly.

2.3.1 Pearson Correlation-based Similarity

Pearson correlation is a technique for finding out the relationship between two continuous variables. To understand this linear correlation between two continuous variables, we can draw a scatter plot of these two continuous variables otherwise it shouldn’t be calculated [10].

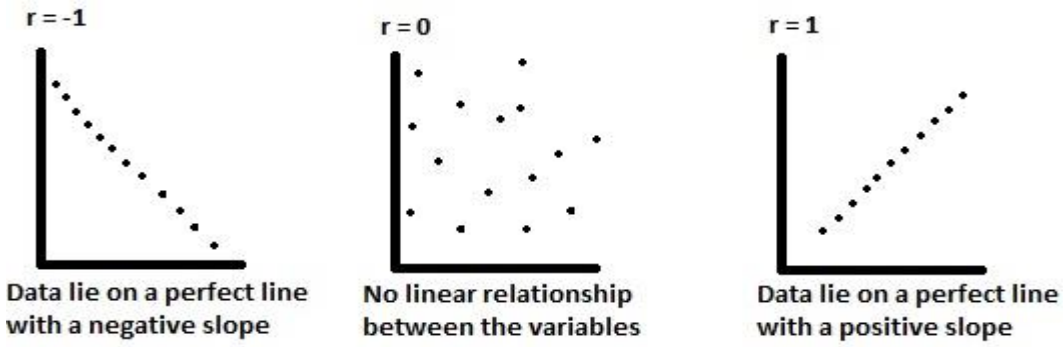


Figure 2.3 Similarity rates according to the each other

So, the Pearson correlation is a number between -1 and 1. “1” means that either variable increase or we can say decrease at the same time, and “-1” means that when one increases, so the other decreases or we can say one decreases, so the other increases. And zero means that there is no relationship between two variables.

2.3.2 Euclidean Distance Similarity

It measures the actual distance between users. Users in recommendation systems are considered as points in a space of many dimensions which are items. In other words, Euclidean distance is the square root of the sum of squared distance between corresponding items of the two users [11]. And as the distance value gets smaller, the similarity of the two users increases.

2.3.3 Cosine Similarity

In this similarity measure technique, items are represented as u-dimensional vectors over user space. The similarity is the cosine of the angle between two vectors. And the cosine score ranges between 1 and -1. If the cosine value is 1 (small angle), it means similarity is perfect. In other case, -1 (large angle) means two users are totally different [12].

Table 2.6 Ratings for each movie from each user

	U1	U2
A	0.8	0.45
B	0.4	0.8
C	0.3	0.3

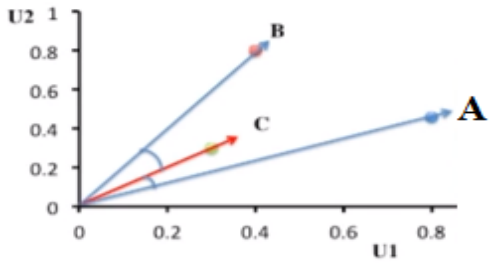


Figure 2.4 Vectors for each user

For instance, as indicated above, we have two users (U1 and U2) and three movies (A, B, and C). If we want to use cosine similarity for computing similarity between two items, the algorithm will look at the angle between two movies. If the angle between the two movies is smaller than the other, it means they are more similar. As in our example, the angle between movie A and C is smaller than the angle between movie B and C. So, we can say that movie A is more similar to movie C than movie B.

2.3.4 Spearman Similarity

It is a type of Pearson correlation similarity measure. In recommendation systems, it tries to find the least preferred item by the user. It gives “1” as the preference value for

this item. After that, it executes the algorithm again to find the next least preferred item. In the same way, it gives “2” as the preference value for the second item, and so on. Lastly, it uses the Pearson correlation to compute the similarity on those converted values. Actually, because of expensive calculations, Spearman correlation is not preferred much.

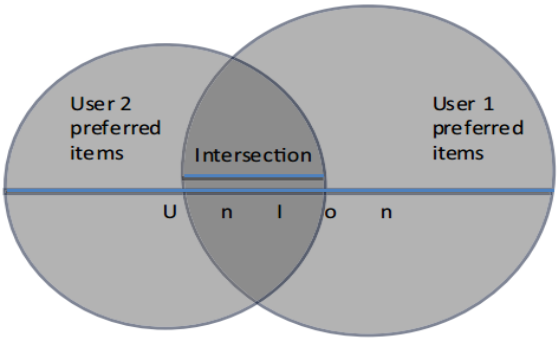


Figure 2.5 Illustration of Tanimoto coefficient [9]

2.3.5 Tanimoto Similarity

Instead of considering preference value whether is high or not for an item, it takes into account that the user has a preference or not to that item. It is also known as the Jaccard coefficient. So, the Tanimoto coefficient cares about the ratio of the size of the intersection between two users. That is how many items that are preferred by those two users. Hence, if the two users’ items are exactly the same, in other words, if they completely overlap, it means that they are totally similar [9].

2.3.6 Log-likelihood Similarity

Its logic is similar to the Tanimoto coefficient or the Jaccard similarity. It does not also care about preference value whether is high or not. There are a couple of problems in the Tanimoto coefficient when computing the similarity ratio. For Instance, when both two users have only seen several movies which are the same movies, the ratio of the size of the intersection between these two users will be high. But are they similar? Because of this kind of problem, Log-likelihood tries to compute the overlap between

users without a chance. Although it looks for the number of items in common between two users, the log-likelihood is computing the similarity a bit different [9]. Namely, it checks dissimilarity between two users as well.

2.4 Challenges in Collaborative Filtering

To better understanding the aims of the research and compare the collaborative filtering recommender with other methods, we should take a quick review of the challenges in CF recommendation systems such as “cold start problem”, “sparsity problem”, “scalability”, “overspecialization”, “robustness”, etc.

2.4.1 Cold Start Problem

The cold start problem defines that the recommender does not have sufficient similarity measurements or ratings to make some product or service recommendations to the target user. Naturally, as long as these values do not exist, or the absence of these values increases, the operating performance of the CF recommender gradually decreases. In general, three scenarios are generally emphasized in the realization of this situation.

New User Problem: This happens when a new user registers on the respective platform. In order to CF recommender advise the target user, at least some items are expected to evaluate or experience by the user. Thus, the CF recommender can find the users that are similar to the target user or products that are similar to the target user's preferred items. However, when a new user subscribes to the system, the CF recommender cannot calculate efficiently and cannot make recommendations. A few solutions are proposed to overcome this problem [13]. For example, the user can rate certain products or services without using the system, or providing non-personalized recommendations until the user is able to spend enough time on the system and score enough products, or asking the user directly about some information like "What kind of movies do you like?", or by providing demographic information within the capabilities of the system, to make some recommendations through similar users.

New Item Problem: Just as in the new user problem, when a new item is entered into the system, the CF recommender cannot make the necessary similarity measurements since it is not experienced by any user. Since the added new items are difficult to

discern by the system users, the CF recommender suffers from the advice of these new products. Different methods can be tried to prevent this situation. For example, it is advisable to present and experience the newly added item on the home screen or make use of the content-based recommender to calculate the items that are similar to the content of the new product.

New System Problem: The biggest and most complex cold start problem occurs when a new system is installed. In this case, the CF recommender cannot work because there is no information about both users and products. To overcome this situation, either a sufficient time is allowed for users to use the system and then CF recommender is used, either a small group of active users is encouraged to rate items in the system, or wait until sufficient data is generated using non-Collaborative Filtering recommenders.

2.4.2 Sparsity Problem

One of the problems that the CF recommender suffers most is the sparsity problem. Users that have not rated many items especially in large datasets give cause for empty cells in the user-item matrix. The CF recommender cannot work fully efficient in calculating user or item similarities before these empty cells are filled. As a result, the performance of the CF recommender decreases. To overcome the sparsity problem, there are a couple of approaches such as one of the techniques of dimensionality reduction, Singular Value Decomposition (SVD), or Latent Semantic Indexing. For instance, SVD actually uses an intenser user-item matrix that includes only the most relevant users and items [14] and removes insignificant users or items in order to decrease the dimensionalities of the user-item rating matrix. Some recommender systems use content-based filtering with CF filtering together to overcome the sparsity problem. Because content-based filtering uses the attributes/features of the items which don't need to rate. But these techniques do not always enhance the performance of the recommender systems, even sometimes may affect their performance worse.

2.4.3 Scalability Problem

Another problem associated with recommendation systems is scalability which means that "*how quickly a recommender system can generate a recommendation and the*

second is to ameliorate the quality of the recommendation for a customer" [15]. A good recommender system is expected to continue to operate in spite of rapidly increasing users and items in the dataset. Although the recommender system initially offers quick and effective recommendations, if it starts to stumble with an increasing volume of a dataset, it means that it suffers from scalability and decreases the quality of the recommender system. The scalability problem can be resolved by cleaning noisy data with pre-processing and clustering. As in Sparsity, SVD can also provide a solution for scalability, although it requires expensive matrix operations. In addition, this problem can be solved with item-based CF to some extent. That is, rather than calculating all similarities between all product pairs, similarities can only be calculated with the items co-rated by the target user [16].

2.4.4 Overspecialization Problem

Overspecialization problem occurs when the CF recommender system recommends only items that have a high score based on sales or ratings [17]. This means that the recommendations which are similar to the products in the user's profile are repeated in the same order. In this case, the user will be constantly exposed to items with the highest likelihood of his/her profile. A good CF recommender should be able to recognize such situations and create alternatives. For example, if the user does not click on the recommendation items list for a certain period of time, CF recommender can change the list according to the order, it can filter the items recommended continuously, present similar products randomly or offer different alternatives with the help of other recommenders such as content-based recommenders. For the sake of example, while the user likes drama movies, CF can also recommend different types of movies with the characters he likes in the movies he watches (with the help of the content-based recommender). To give another example, a user who is constantly exposed to recommendations from Italian cuisine will not be aware of Greek or Turkish cuisine. However, the food used by these countries in the Mediterranean band is likely to be similar and food tastes are also likely to be similar. That is, the recommender should notice some points that the user cannot see in the system and make the user realize the other options. Otherwise, users may be constantly exposed to the same recommendations by the CF recommender and this situation may bore users after a

while. This problem is more common in the content-based recommender. Because content-based recommender rates the items according to their content and features, each time an item is searched according to the user's profile, the relevant items that cover the most sought-after content will come to the forefront.

2.4.5 Robustness Problem

In today's e-commerce environments, the usage of recommender systems is increasing day by day due to the convenience provided to the users. But as e-commerce began to compete with the traditional trade, the value that the business world gave to recommender systems began to increase. With this increase, the number of fake accounts who want to take advantage of this situation has also increased. Namely, the number of malicious users, biased bloggers, and even the owner of the products that are trying to influence the systems have started to increase day by day. Malicious users who cannot get the service they expect, open multiple accounts and run a smear campaign on the relevant platform, while biased bloggers sometimes try to inflate a product or a service most of the time, sometimes scribbling a product or a service, sometimes for money, sometimes completely for pleasure. In addition to that, some users are the owner or provider of the product on the relevant platform, they open more than one fake account and give high scores to their products or offer positive opinions about their services. When such product or service owners swell the scores of their products, we call push attacks, to reduce the score of competitors' products, or to make scribbles of their services, we call nuke attacks. According to the research results [18], in order to overcome this problem, using item-based CF recommender that is thought better than user-based CF recommender, observing the results of the recommender and check for sharp changes, observing whether the newly added product to the data set is obtained from a trusted source or whether it is evaluated by reliable users, are the some suggested solutions.

3. IMPROVING THE QUALITY OF RECOMMENDER SYSTEMS THROUGH THE TRUST RELATIONSHIP

In this chapter, we propose a trust-based method for improving the quality of the recommender systems in e-commerce platforms. The proposed method is expected to provide a certain extent of the solution especially to the sparsity and robustness problems mentioned as in the previous chapter. Therefore, in order to overcome especially negative effects of the fake accounts in e-commerce platforms, the proposed recommender model finds the users who are trusted and have a great effect on other users' opinions by analyzing the relationship between them. With the proposed model, recommender systems are expected to provide better recommendations to users based on trusted users' ratings.

The greater number of the customers shopping online increases with developed secure e-commerce systems, the more companies start to work on this field. Moreover, it is speculated that the amount of commerce done on e-commerce systems, will soon pass the amount in the traditional commerce [19]. But to be successful in this field, it is necessary to determine customers' behaviors for improving the quality of recommender systems. For this purpose, e-commerce companies of today analyze the click and purchase history of users or customers. Unfortunately, the feedback of users is insufficient for analyzing customers better. Even most of the companies state that the density of their database is less than 1% [7]. This is really a major obstacle in front of the further success of the companies.

Today, when buying a product online, the product score is very important when making our last decision but due to the sparsity problems in databases, fake accounts can easily affect results of recommender algorithms especially when the product doesn't have enough votes by consumers. Generally, fake accounts are created either by the owner of the product in order to raise their product score or by the ill-wishers who want to denigrate a product or a company. For instance, on average 100 fake accounts can easily identify the score of a hotel on TripAdvisor if that hotel does not have too many votes. Thus, in order to overcome the negative effects of the fake accounts in e-commerce platforms, we try to create a recommender model finding the users who are trusted and have a great effect on other users' opinion.

On the other hand, some products or items do not have enough ratings in order to calculate their real rating value or score. Sometimes items get very high or very low rating value because of this reason. Therefore, we can get the real rating values of items by looking at trusted users' opinions. The relationship between customers is revealed via the PageRank algorithm in order to find out trustful customers and recommendations are provided to customers based on those trusted users.

The significance and implications of the proposed method can be listed as follows:

-It is a different recommender model which is based on trusted users' ratings, unlike the current recommender systems.

-It breaks down the power of the fake accounts on recommender algorithms in order to get a real score of a product.

-It helps to overcome the sparsity problem of recommender systems in e-commerce platforms.

-The system is based on trust relationships between users, but it is different from the existing trust-based recommender models since the trustworthiness value of a user is calculated by the consensus about an item not the similarity between target users to others.

-It can be used for comparing with the traditional average score when buying or consuming an item to confirm the quality. This is really a big problem for the users because they generally buy the items by trusting the average score, and most of them are boomed values.

Before starting to explain the background of the proposed method, it would be healthy to have a look at the recent studies conducted in the areas of CF recommender and trust-based recommender, especially for comparison with the proposed study.

3.1 Related Work

In this section, to better understand the subject, we present the recent researches in the field of CF and trust-based recommender under two separated headings.

3.1.1 Studies in Collaborative Filtering Recommender Systems

Traditional CF recommenders present some platforms to users analyzing their ratings, clicks, purchase histories, the relationship between items and users, or demographic information on the system in order to provide preferable products, information, services, people, etc. But nowadays, it is a necessity to develop personalized recommender systems due to the ever-increasing product range and intercompany race in e-commerce platforms, social networks, and search engines, etc. Therefore, in this part of the section, we review some related studies especially on social network-based recommenders that largely depend on one of the demographic information, influential ranking algorithms, content-based filtering, and especially collaborative filtering.

Trujillo et al. [20] carried out research in order to work up the performance of the recommender system based on multi-features such as demographic and psychographic information to calculate the similarity between users. Demographic information defines the user's information such as age, gender, education, etc., while psychographic information defines user's interests and documents downloaded by the user. The similarity between users is calculated adding firstly based on demographic features, secondly based on interest areas, and lastly based on downloaded documents, respectively. According to the results, in order to cope with the main drawbacks of the CF recommender, all this information should take into account when providing recommendations.

In order to overcome some challenges that every recommendation system suffers from such as sparsity, scalability, and prediction accuracy, Ma et al. [21] presented a novel approach called SoRec (Social Recommendation) integrating users' social network information with rating matrix. According to the authors, users are affected by their social connections, and recommender systems can deal with missing values using these relationships efficiently. In order to use users' social network information with a user-item rating matrix, the authors use the conditional distribution on the social graph and the conditional distribution on ratings. According to the authors, trust value decreases when the target user trusts lots of users and it also increases when a user is trusted by the lots of users.

In order to increase the performance of recommender systems, Shin et al. [22] proposed a context-aware recommendation system by clustering context information of a user. In

order to achieve this goal, firstly the authors obtain raw data and then they resume raw data to the conceptual level. After that, they aggregate user conceptual context information to create a better recommendation. According to the authors, the time of consuming an item is an important factor for the recommendation system. User preferences can be changed according to the time of the day or the day of the week or the season of the year. To provide a better-quality recommendation, firstly the authors calculate the similarity between the current context of the target user and clustered context by using cosine similarity. Secondly, they calculate the similarity between an item and clustered context and lastly, they multiply both results together in order to get the expected preference of a user for an item. Results verify that the performance improve with only context information if it is compared to conventional recommendation approaches.

Jamali and Ester [23] analyzed and compared the importance of social influence network and similarity network in order to get rating prediction. They investigate whether user rates after exposed an item rated by the target user's neighbors in time or not. The authors explored both item adoption and ration adoption in social network and similarity network applied on Epinions and Flixster dataset. According to the experiment results, the influence of neighbors (direct neighbors) or rating items in the social network is higher than in the similarity network on both datasets when the user exposed to an item at a time.

In order to provide better-personalized recommendations in social tagging systems, Zheng and Li [24] proposed a new computational approach using tag and time information. For this purpose, the authors use three strategies. The first one is the "tag weight" strategy aiming to compute the weight for every resource using users' tag information. The second one is the "time weight" strategy for computing the weight of interest drifts for every selected resource by the target user. The last one is the "tag and time" strategy for calculating the target user's rating values with the combination of tag and time information. According to the experimental results, all three strategies are effective and better than the log-based approach.

Due to suffering from just using one algorithm in the recommender system, Yu [25] proposed a new method called dynamic competitive recommendation to use in social networks. For this purpose, Yu present an algorithm that calculates the recommendation

with several algorithms. The author carries out the experiment on the Twitter platform in order to recommend a friend to the target user by calculating several component algorithms such as “Number of Followers”, “Number of Tweets”, “Common Follower”, “PageRank”, and “Profile Matching”. Consequently, the dynamic competitive recommendation algorithm chooses the highest one according to the results of each algorithm. If the results are equal, the dynamic competitive recommendation algorithm tries to get the results from each algorithm according to the maximum, average, Standard Deviation, and size of the candidate list, respectively.

Kim et al. [26] propose an enriched collaborative user model and rather than clustering users in accordance with the topics, they use a topic-driven user model. For this purpose, they integrate ratings and tags to discover frequent topics not only relevant to user interests but also irrelevant. According to the ratings given by the target user, the algorithm determines tags that the target user would be interested in or not. In this way, the algorithm gathers tags, if these tags are frequently annotated in positive or negative items; it means the target user is interested in a particular topic or vice versa. The authors identify the neighbors based on tags labeled in items. According to the experiment results, the proposed collaborative user model is better than other CF recommenders.

Because of difficulties to keep track of different social networking sites for a user, Zhang et al. [27] present a system or application called Social Connect (SocConnect) for getting social activities published by user’s friends in different social networks. In order to achieve this goal, the authors design an application in which the user can select his/her friends from Twitter and Facebook, and also users can make some special groups using tags based on his/her relationships. In the proposed system users are allowed to give some ratings on his/her friends’ activities such as “like” or “dislike”. In this way, the algorithm finds out the target user’s interests in activities using some machine learning techniques. Consequently, Zhang et al. create a personalized system that allows users to blend and grouping their friends as well as tagging their friends and social activities obtained by different social networks domain and provide users some recommendations based on their interests from the related social networks.

Ullah, Sarwar, and Lee [28] offer an interesting study of the use of recommender systems in a different area. The authors propose a smart device that recommends TV

programs according to user's preferences and social network data. The authors convert user's preferences to a rating value between 1 and 5. To calculate the rating value, they divide the time which the target user spends on the program during the broadcast, with the total time of the program. After getting the rating value, the authors try to find the features of the related program such as genres, actors' information, etc. Then, they select the top-N most similar users. To find a similarity between two users, the authors calculate 3 different parts. The first part is the Pearson Correlation Coefficient similarity metric. The second part is direct trust between the users and the last part is the contribution of the user content. According to the simulation results, the proposed system performs well in terms of accuracy, precision, and recall.

Sun et al. [29] suggest an approach that integrates the social network graph to improve the prediction accuracy of recommender systems using a bi-clustering algorithm finding the most suitable group of friends. According to the authors, when we do not separate the different friendships between users, all the friendships will be treated equally. But in this way, we can't improve the accuracy of the recommendation. For this reason, Sun et al. cluster the dataset in order to get smaller groups with similar favors. The experiment on the real dataset reveals that if we do not ignore the friendship among users, we can improve the prediction quality.

Yu-sheng et al. [30] propose a model called interest social recommendation (ISoRec) consisted of a combination of the user-item matrix, implicit user interest information, and explicit user social information. To build up a new model, the authors combine the following two basic models. The first model is named social recommendation (SoRec). The model uses the user-item matrix and implicit interest relationships matrix simultaneously. Because of reflecting the interest of the user's friends' interest, the second model is named the social trust ensemble. Namely, to improve the prediction accuracy of recommender systems, it uses the users' explicit social connections. According to the comparing results, the proposed model outperforms PMF (Probabilistic Matrix Factorization) and SVD on the MovieLens dataset with respect to RMSE and MAE. The proposed model also outperforms the SoRec approach on the Epinions dataset. But the authors use implicit users' interest connection information when the explicit social connection information is not available. Actually, this is the main idea in the related study.

Han et al. [31] propose a prediction model in order to find the interests of a user who doesn't have enough information to the recommendation in an online social network. For this purpose, the authors utilize social information such as demographic information (age, gender, current city, etc.), social relationship (user friend list, friend similarity), and obtainable users' interests (interest entropy). According to the proposed solution, to calculate users' interest similarity, firstly we can compare and measure the geographic distance between users, secondly, we can count the mutual friends of two users, and lastly, we can employ entropy to determine a user's interest feature. In order to calculate the interest similarity, the authors use binary similarity measure and weighted cosine similarity measure (if the mutual interest is too common, it has less effect such as the movie "Harry Potter"). In the research, the authors use leave-one-feature-out evaluation to see the effects of each social feature on the interest similarity prediction, but they see that all the used features have an effect on the prediction. Since, each social feature has an importance on different domains such as location has a great effect on music similarity measure than movie similarity measure, while social relation has a great effect on movie similarity measure than music similarity measure. At the end of the research, the authors compare the proposed prediction model with several state-of-the-art recommendation models for a new user. According to the experimental results, the proposed prediction model is better than other models with a great difference.

Yuan et al. [32] show that friends have different influences on users' behavior in the social network. Some of them, called "buddy" in the research, have a strong influence on a user. According to the authors, some users also are not influenced by other users when they make a decision. For this reason, in addition to finding buddies, Yuan et al. calculate the susceptibility of each user by the social relation analysis. So, the recommendation is done based on both friends' influence and individual taste. To find the user's closest friends and calculate his/her susceptibility, the authors use rating similarity and edge embeddedness. If the target user and his/her buddy have high taste similarity and common friends (edge embeddedness), it means that the buddy has a great influence on the target user. The experiment is conducted on Douban and Epinions' real-world dataset which contains the users' rating and their social relationships. The proposed algorithm (BSSR) is compared with other algorithms such as ItemCF, PMF, SoRec (Social Recommendation), RSTE (Recommendation with

Social Trust Ensemble), etc. According to the results, the proposed algorithm model gets remarkable progress and can be better to use on social relations into the recommendation systems.

Chaney et al. [33] present a probabilistic model called Social Poisson Factorization (SPF) combining user preferences for items in traditional recommendation systems with social networks influence information developing a scalable algorithm. According to the authors, when we decide to choose something, our behaviors are affected by our general preferences and influence of our closest friends. The computation of "influence" in the research is not the same as trust information. Because trust information is calculated by a binary structure (if it has a link between users, the trust value is equal to "1", or "0" otherwise) and it is computed on the structure of the network without considering user activities and similarity degree between users. But SPF calculates the influence by looking at the similarity between the target user and others in addition to user behavior history. According to the results, SPF achieves top performance on five different datasets against the competing methods such as social factorization, poisson factorization, popularity baseline, etc.

To improve the quality of personalized recommendations Gan [34] proposed a novel method called COUSIN which is a network-based regression model correlating object and user similarity profiles. For this purpose, Gan creates two matrices consisting of user similarities and object similarities using historical data. Then in order to obtain a sparse similarity network, he carries out the power-law adjustment on these two matrices. After that, he creates two concordance vectors consisting of "user similarity profile" and "associated object similarity" obtained sum over similarities between selected objects by the user and candidate object. To obtain the "associated object similarity profile", this process is repeated for all the users. Lastly, to calculate the concordance score, Gan applies a regression model on these two vectors. According to the result, the proposed method shows better performance over existing methods not only the accuracy but also the diversity of recommendations because the method uses both user relationships and object relationships in a single regression model.

In order to improve the effectiveness of recommendations, Colace et al. [35] proposed a novel collaborative and user-centered recommendation approach using some aspects related to the target user such as preferences, opinions, behavior, feedbacks integrated

with item features and content information. For this purpose, the authors cluster items based on similarity considering all the different features. The authors filter out the set of features browsed by the user. Namely, users are defined by the set of features, not by items. The main idea of the research is to find similar items that the target user browses in the same session. So, if the target user browses an item o_i after an item oj, the algorithm marks both items positively for rating. The authors also use the Mixed Graph of Terms (MGT) for refining items ranks with target user sentiments and feedback. The MGT includes some interrelated words which describe a certain sentiment belonging to a knowledge domain. The results show that the proposed approach can be easily used in some types of platforms to provide recommendations for more than one category of items.

To provide better recommendations, Celdrán et al. [36] present a hybrid recommender to rank the suggested items by combining similarity between user to user, user to items, items to items, and location of users and items. Firstly, the authors filter undesired items by looking at the properties of a given item. If any properties do not intercept between the item and target user's preferences, the algorithm filters that undesired item. In order to compute the similarity between user and item, the authors care about the number of common properties between item and target user preferences. They compute the item similarity looking at the number of common properties between the given item and all the items chosen by the target user without considering the importance of properties. To compute user similarity, the authors combine the items visited by both users and the ratings of two users with similar preferences. To compute the location of users and items, they use the Manhattan distance to measure the distance between users and items. To compute the users' tracking, the authors calculate the number of times of visiting and the direction of the user to recommend items on that location and lastly the date when the target user visited the related item last time. The experimental results show that the proposed solution is useful and efficient.

Yang et al. [37] present a hybrid model in order to predict better how a user similar to one another. For this purpose, they use demographic information such as age, gender, and location and they use social network information such as friendship, relationship, and group information. To represent this goal, they carried out the experiment on the video domain using a tag-based user profile. The authors also combine that information

in order to predict user similarity based on different machine learning techniques. The authors use a popular Tag-Based Profile which is based on the frequency of common tags among users and Representative Tag-Based Profile which is based on tags representatives. The authors also take into account the time of the target user's interest. According to the authors, the performance of the recommender system is changed as regards the time of the user selecting an item. Namely, the current interest of a user is more effective on the performance of the recommender.

Biancalana et al. [38] propose a hybrid recommender system that calculates contextual factors related to time to improve the quality of collaborative filtering approaches. According to the result, new items (which are movies in the study) can have a higher potential of being interesting than old ones.

Fijałkowski and Zatoka [39] present a concept to improve the effectiveness of e-commerce recommender systems using social network user profiles obtained via facebook Open API. The authors propose to obtain some objects from user's facebook profile such as user posts, published links, comments along with user's friends' posts, and comments liked by the user. These objects can be used to calculate similarity between users' interests based on keywords list in the context in order to enrich dataset in e-commerce platforms.

Carrer-Neto et al. [40] use an application based on movies. According to the experiment results, adding social heritage to the recommender system decreases the quality of the recommender since more contents affect it.

Bedi et al. [41] present a recommendation method using knowledge domain for generating a recommendation. According to the selected domain by the user, the algorithm brings the product based on trust calculating by that chosen product experiences. When a user selects a product, the related domain of the target user is changed by the system. So, the trust means, in the research, how many times the users selected that product, in other word it is based on experiences.

3.1.2 Studies in Trust and Reputation Based Recommender Systems

Because of providing remarkable improvements, many researchers have been investigated trust-based recommender systems on social networks in recent years. The basic idea is that our preferences are not completely independent from our relationships.

Namely, when we think of buying or choosing a product, our friends' opinions will have a significant influence on our decisions. In this section, we review several major approaches for trust-based recommendations in general.

In place of using similarity between users' profiles in CF to calculate the ratings of an item, O'Donovan and Smyth [42] proposed a new approach based on the trustworthiness of users on a specific rating prediction. In order to achieve this goal, the authors modified a lit bit the Resnick's prediction formula. Due to the fact that Resnick's prediction formula gets rating prediction by looking at profile similarity, the authors propose to add trust to the formula. Their algorithm calculates the trust by looking at the percentage of correct recommendations comparing predicted ratings between the target user and any other user. If the user has a greater number of correct recommendations with the same target user, the algorithm gives more trust value to that user. The algorithm also filters some users by looking at the trustworthiness degree by using a threshold. According to the results, the use of trust value has a great positive impact on overall prediction error rates.

To improve the trustworthiness of rating prediction, Jamali and Ester [43] proposed a random walk recommender model, called TrustWalker, using both trust-based and item-based collaborative filtering recommendations. In order to provide a recommendation to a user, the authors apply random walks on the trust network. If the user has already rated the target item, the random walker returns with that result, otherwise, it has two options to continue. The first option is to stay at that node and select one of the items similar to the target item and return with that value as a result. The second option is to continue to another trusted user. The results show that trust-based CF performs better than other CF methods to cope with fraudulent attacks.

To construct an efficient and effective recommender system, Ma et al. [44] presented a novel approach called Social Trust Ensemble combining the user's preferences with trusted friends of the user. In order to fuse users' social network information with the user-item rating matrix, firstly the authors calculate conditional distribution on ratings (if the user rated the related movie, it is equal to "1", or "0" otherwise). Then they calculate the conditional distribution on a social graph which is not symmetric, namely, if the user "u" trusts user "k", it doesn't also mean user "k" trust user "u" and they use the weighted trust edge between two users. They also use a balancing parameter "α"

controlling the users' preferences and the trusted friends' preferences. So, when "α" is equal to "1", it means recommendation will use just user's preferences, while "α" is equal to "0", it means recommendation will use social trust network in order to generate a recommendation. According to the experimental result, when the "α" is equal the 0.4, it is the best condition for a recommendation. In addition to that, results show that when the recommender uses just the social trust network, it is worse than just using users' own tastes.

In order to improve the prediction accuracy of recommender systems, Lathia et al. [45] propose a variation of a k-nearest neighbor algorithm to reveal how much a user close to the target user in the recommendation system. For this purpose, firstly the authors use the traditional Pearson Correlation Coefficient in order to find the similarity between two users. When calculating the rating prediction, the algorithm gets all users who rated that item. The trust value is increased when the distance between two users' rating decreases and the trust value ranges from 0 to 1, whereas the similarity value ranges from -1 to 1. After completed trust values for all users, the algorithm gets some of them according to the k-Nearest recommenders which select the user who provides necessary information for rating prediction for that item. According to the results, the proposed method outperforms the similarity-based methods to cope with problems in CF.

Hang and Singh [46] proposed a trust-based recommendation considering link structure and trust network. In order to calculate the similarity between graphs, the authors use a convergent iterative process. The method is applied to a vertex similarity measurement between graphs by calculating the similarity between the trust network and a structure graph. According to the results, the similarity measurement between the trust network and a structure graph shows how tightly a user connected to his neighbors.

In order to increase the prediction accuracy of recommender systems in e-commerce, Li et al. [47] use the Multi-Criteria Decision Making method that consists of preference similarity, recommendation trust, and social relation. To predict a recommendation, the algorithm should calculate all three categories. According to the experiment results, the STR model (preference similarity, recommendation trust, and social relation) has higher success frequencies than others when "ε" (the absolute value of prediction error) is smaller than 0.6 and has lower success frequencies than almost all others when "ε" is larger than 1.2.

To improve the prediction accuracy of recommendation systems Chen et al. [48] propose a novel approach based on social trust relationships called RSTR that explicitly and implicitly uses social trust relationships simultaneously. Actually, this study is a combination of SoRec proposed by Ma et al. [21] and RSTE proposed by Ma et al. [44]. The main goal of the research is to estimate the missing data in the user-item rating matrix using a social trust relationship between users. SoRec algorithm indicator function gives "1" value if the target user trust to another user or "0" otherwise and likewise, the indicator function gives "1" value if the target user rated an item, or "0" otherwise whereas in RSTE algorithm both target user's and his trusted friends' preferences affect observed item rating. According to the results, the proposed method combines the advantages of SoRec and RSTE.

To improve the quality of collaborative filtering recommendation, Yang et al. [49] propose an approach called TrustMF by way of compounding ratings and social trust network. With this object in mind, the authors use a trustor-specific feature vector and trustee-specific feature vector. The main mentality of this trust-based study is one can be affected by other users, and one can also affects to other's opinion. Therefore, to calculate a user's ratings affected by other users, the authors use a trustor model. In order to get the value of a user who affects other's decisions, the authors use a trustee model. Results show that TrustMF performs better than its competitors for the cold start problem on a real-world dataset.

O'Doherty et al. [50] presented an empirical analysis to compare trust-based recommendation algorithms. In the research, the first algorithm is like Pearson weighted mean just instead of the similarity measure, the algorithm uses trust value between two users. The second algorithm is also like Pearson collaborative filtering just instead of the weighted similarity measure, the algorithm uses trust value between two users. The third algorithm is like a simple mean but when the algorithm calculates the rating prediction, it just admits the raters who pass the trust threshold. The fourth algorithm is also like Pearson collaborative filtering but when the algorithm calculates the rating prediction over similarity, it just admits the raters who pass the trust threshold. The fifth algorithm goals consider all available ways to get a positive weight for a user of an item when verifying trust value over similarity. According to the experiment result, accuracy and coverage of trust-based algorithms are better than standard algorithms.

To improve the effectiveness of the recommendation system, Özsoy and Polat [51] compare existed recommender systems and propose an approach using a trust-based recommender system. They use a set of users, items, tags, and categories. The items are defined by tags and categories also have related tags. In order to find the probability of liking an item by the target user, firstly the algorithm controls the ratio of the number of items commented by the target user posted by a neighbor-user over all items commented by the target user in that specific category. Then the algorithm calculates the ratio of the number of all items commented by the target user over all items in that category. Lastly, it calculates the ratio of the number of items posted by the target user over all items in that related category. According to the results, using combined trust-based recommender systems performs better for providing personalized services rather than the similarity-based method.

Deng, Huang, and Xu [7] presented an approach to service recommendation using a trust relationship between social network users called as RelevantTrustWalker (RTW). To measure the trustworthiness degree between users, the authors use a matrix factorization. Then, they get recommendations results by the use of RTW which is a random walk algorithm based on trust relevancy among users. In order to predict ratings, RTW chooses the neighbors according to the weighted trust social network. Trust relevancy is calculated multiplying similarity and the degree of trust of the user "u" to the user "v". Because the random walk is likely to never stop, the authors choose 6 degrees for the maximum step. When the RTW chooses the user, it tries to find the rating of the related item giving by the chosen user. If the chosen user has rated the related item, that rating is returned as the result of that walk. Otherwise, the algorithm goes on to the next user who is again chosen by the trust relevancy formula executed on the last chosen user. If the next user has also not rated the related item, the algorithm gets the rating of the most similar item to the related item. So, the RTW gets the ultimate result through multiple iterations. Results show that the proposed method chose the target node not randomly but based on trust relevance when a random walker tries to find a similar user.

Instead of calculating the user similarity method in order to predict the rating of an item, Zhong et al. [52] propose to use a directed trust graph. For this purpose, firstly the authors reveal directed and undirected relationships between users. Thus, when

calculating the predicted rating, the algorithm takes into account if the target user has a direct or indirect relation to the other users. In order to achieve this goal, the authors use "direct and indirect trust nodes" together. Namely, there is a level of distance from the target user node to other node and according to this level of distance, the result is changed. If the neighbor node is in the first level (direct relationship), it affects the result more than other levels. So, when the level increases, the affection of that node on the rating value decreases. The experimental results indicate the good effects of the proposed trust-based recommendation approach.

Chamsi et al. [53] propose an approach to transform concepts of users' profiles from social networks into a source of recommendation. They crawl the user's tweet, retweet, and replays via Twitter API in order to get concepts for each user. After applying cleaning operations, the authors try to find the frequency of each concept appeared in each user profile. In this way, they build a matrix of concepts for all users. Chamsi et al. apply the same filtering operation over the resources to build up a matrix of concepts for all resources. The authors use the memory-based collaborative filtering and apply a user-user based recommender algorithm to get the predicted value for the target user of an item related to a concept. Consequently, the authors use social information as a resource for the recommendation system to offer a resource to Twitter users. Results show that transforming the social network into a recommender can be useful.

Guo et al. [54] carried out an empirical study in order to compare five different trust algorithms on two different datasets. In order to achieve this goal, firstly they present five trust definitions used in recommender models by taking into consideration four trust properties. These four trust properties are Asymmetry (trustworthiness is mutual or not), Transitivity (if user A trusts user B, user B trusts user C, then user A trusts user C), Dynamic (trustworthiness changes in time or not), Context-Dependence (trustworthiness bases on knowledge domain or not) respectively. According to the empirical study results, there is no single trust algorithm superior to others when the data set is changed.

In order to personalize recommendations, Alahmadi and Zeng [55] present a new approach using trust relationships and users' friends' comments crawled from Twitter. The authors calculate the trust value between the target user and his friend by normalized average "RT" (re-tweet action means how many times the target user re-

tweet the message of his friend over all the messages which are re-twitted by the user "u" from all his friends) and "L" value which indicates the percentage of followers the overall number of followers and followings. According to the results, short and informal posts published by users in social data can improve the quality of the users' preferences data, especially for CF problems.

In order to lead to improved predictive accuracy during recommendation, Deng et al [56] present an approach called Trust-based Service Recommendation using preferences of users and trust relationships among users by looking at invoked services by each user. For this purpose, firstly the authors prepare "History Service Records" which consist of services called by the active user, categories according to the service domain, and the rating value given by the active user to the related service. According to the trusted user set, the authors calculate the similarity between two users and provide the top-k services to the target user using those similarities. According to the experimental results, the proposed method performs better recall rate, precision, f-measure, and rank score.

In order to increase the prediction accuracy of recommender systems, Keikha et al. [57] proposed a method called TB-CA (trust-based context-aware) using the information of a user trust networks to recommend items that are matched user preferences. For that purpose, the authors firstly apply a few preprocessing on raw data in order to get conceptual context data by Fuzzy function and they form the set of trust networks of each user. Then with a given user and target items, the algorithm starts to find out the rating of the target item. Firstly, the algorithm looks at the user's item set which is formed with the same conceptual domain of the target item. Namely, if the target item is in that rated set, naturally the rating of the target item is returned. Otherwise, a random walker selects a neighbor of the target user and the same process is repeated. According to the results, the proposed method is successful with all contextual concepts and F-measure compared with previous methods.

To improve the quality of item recommendations in social networks, Wu et al. [58] set up a new algorithmic framework called collaborative topic regression (CTR) with social trust ensemble exploiting user-item feedback, item content, and social network. The authors offer the rating prediction to the user "i" for item "j" by looking at user i's and item j's latent features with the social influence which is a weighted sum of the

predicted ratings for item “j” from all of the user i’s trusted friends. Trust score is calculated via similarity between two users with the number of users who “i” follows or we can say “i” trusts and the number of users who trust the user “i”. According to the experimental results, while the social influence is effective on one dataset's users, the individual tastes are more important on another dataset's users without considering different scales of training data. Actually, this situation shows us, the influence of social media on recommendation systems differs from the nature of our data.

Now we will explain the basic mathematical background related to the calculation of product rating score. After that, we will give some basic information about graph theory related to our used formula. Let’s explain how we calculate the rating scores of products based on trust relationship between users.

3.2 Prepare Background and Context

In recent years, with the rise of the importance of recommender engines in e-commerce platforms, people and companies start to affect the results of recommender algorithms in order to increase the rating scores of their products by creating fake accounts. Because, as in real life, when we make our decision about buying or choosing a product or service, obviously, we are influenced by the opinions of the people surrounding us. Especially in e-commerce platforms, people pay attention to the rating of products that they want to buy or use as a service and examine most of all, if not all, the reviews before making a final decision. For example, when we search for a hotel on TripAdvisor, at first, we are checking on how many stars that hotel has, and then we are reading almost all the reviews about that hotel. On these types of websites, people can also click like button if they agree with that comment or dislike button if they don’t. This feature is also important to understand the quality of that hotel. In this section, we analyze the relationship among users to find users, who are trusted by others and calculate the ratings of products according to those trusted users’ ratings. In this way, we can present two types of ratings to the users. The first one is the traditional average score and the second one is a weighted score based on the trustworthiness of the users. Therefore, people will have a chance to compare two rating results. Moreover, it can show the quality of the rating result if the two rating results are almost equal. For this

purpose, we first explain the basic mathematical background of the PageRank algorithm, which we use in our research to reveal the relationship between customers and then find trustful users.

3.2.1 Mathematical Background

Basic linear algebra operations: In math, creating a rectangular array adding m x n numbers in rows and columns is called a matrix. We represent m x n matrix A with real numbers. Namely, it means that someone can get a real number "A_{ij}" on row "i" and column "j". For instance, the value A_{23}, which is pointed "1", is located where the second row and third column coincide.

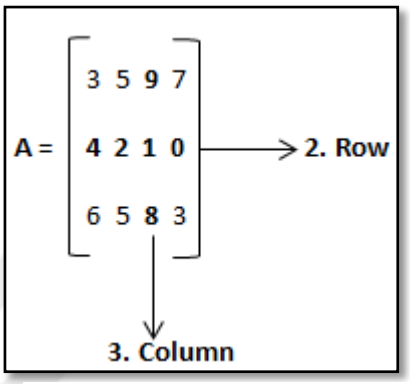


Figure 3.1 Relationship between rows and columns in a matrix

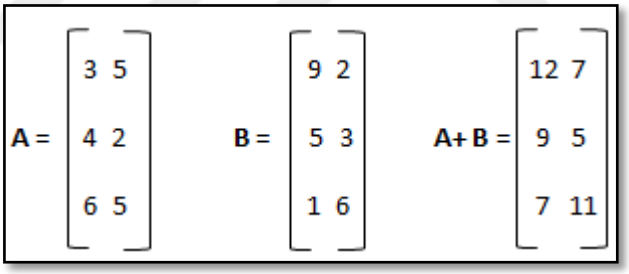


Figure 3.2 Adding and subtracting matrices

We can add or subtract matrices via adding or subtracting the numbers in the same positions if two matrices have the same size. As it is seen in Figure 3.2, matrix "A" and matrix "B" have the same size, namely, they have both the same number of rows and columns. Therefore, we can add both matrices by adding the numbers in the same positions. Furthermore, matrix addition between the same types of matrices has associative and commutative properties.

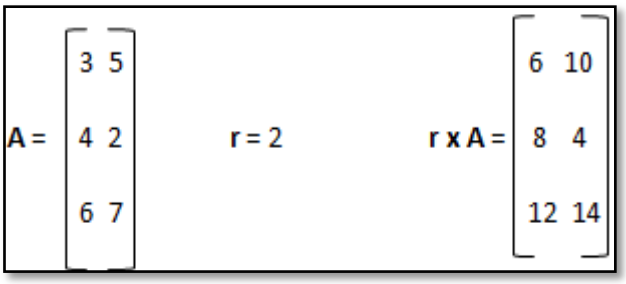


Figure 3.3 Scalar multiplication

As it is seen in Figure 3.3, when multiplying a matrix by a single number (a scalar), each element in the matrix should be multiplied by this scalar. Accordingly, if we have a matrix "A" with single numbers "r" and "k", we can say that (r+k)A = rA + kA and r(A+B) = rA + rB and also (rk)A = r(kA).

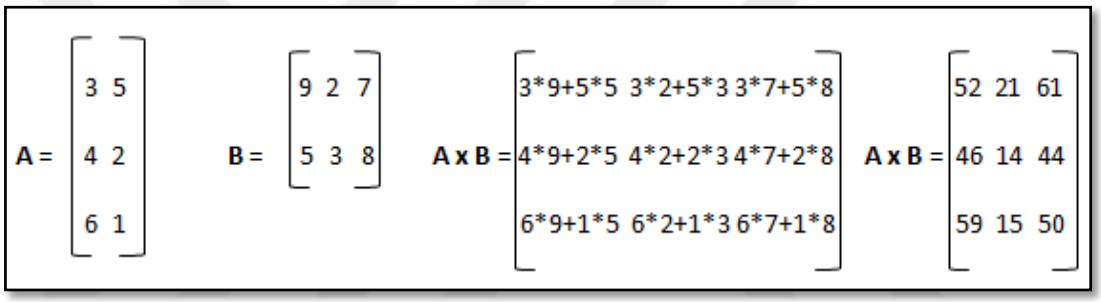


Figure 3.4 Matrix multiplication

In order to multiply a matrix by another matrix, the number of the columns of the first matrix should be an equal number of the rows of the second matrix. If they carry the condition, we can multiply two matrices to each other. Therefore, as it is seen in Figure 3.4, firstly we get the first row of the first matrix "A", then we match the first column members of the second matrix B, multiply them (3 with 9 and 5 with 5), and finally sum them up. Likewise, we do the same procedure for the second column of the second matrix "B" and for the third column of the matrix "B". After finishing each column of matrix "B", we do the same procedure for the other rows of the matrix "A".

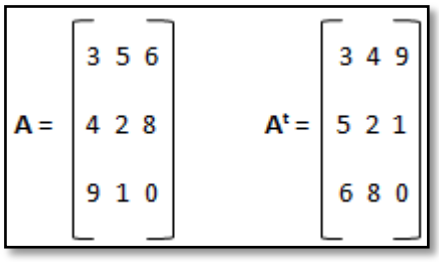


Figure 3.5 Transpose of a matrix

It is called transpose of matrix "A" denoted "A^t" by turning all the columns into rows or vice-versa. So that the first column of the matrix "A" will be the first row of the "A^t", and the second column of the matrix "A" will be the second row of the "A^t", and so on. Therefore, transpose of the "A^t" will be the matrix "A" again. Transpose of two matrix multiplication is equal to the multiplication of transpose of each matrix "$(A \times B)^t$" = $A^t \times B^t$. Likewise, transpose of two matrix addition is equal to the addition of transpose of each matrix "$(A + B)^t$" = $A^t + B^t$.

In a linear transformation T : V → V, a non-zero vector "x" for x ϵ V, if there is a number "λ" satisfying the equality T(x) = λx, there is an "x" called eigenvector corresponding to the eigenvalue "λ" [59]. To sum up this situation with an example:

To find eigenvalue and eigenvector of matrix "A", we should obtain the expansion of det(A-λI) generating second-degree polynomial. $\lambda^2 - 4\lambda + 3 = (\lambda-1)(\lambda-3)$, which is called the characteristic polynomial for matrix "A". As a result, the eigenvalues for the matrix "A" are the solutions of the roots of the characteristic polynomial p(λ) = 0, and they are λ = 1 and λ = 3. To find the corresponding eigenvector of eigenvalue λ = 3, we get a = 0 and b = 1. The transpose of matrix "A" has also the same eigenvalues, but they don't have the same eigenvectors that correspond to the common eigenvalues [60] in general.

$$A = \begin{bmatrix} 1 & 0 \\ 2 & 3 \end{bmatrix}$$

$$\text{Det}\left(A - \lambda \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \right) = 0$$

$$\text{Det}\left(\begin{bmatrix} 1 & 0 \\ 2 & 3 \end{bmatrix} - \begin{bmatrix} \lambda & 0 \\ 0 & \lambda \end{bmatrix} \right) = 0$$

$$\text{Det}\begin{bmatrix} 1-\lambda & 0 \\ 2 & 3-\lambda \end{bmatrix}$$

For example, to find the corresponding eigenvector of transpose A^t for eigenvalue $\lambda = 3$, as it is seen in the example, there is more than one eigenvector corresponding to the same eigenvalue. Even if these eigenvectors have the same eigenvalue, they may have no relationship to each other.

$$\begin{bmatrix} 1 & 0 \\ 2 & 3 \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = 3 \begin{bmatrix} a \\ b \end{bmatrix}$$

$$\begin{bmatrix} 1 & 0 \\ 2 & 3 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} = 3 \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 2 \\ 0 & 3 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \end{bmatrix} = 3 \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

After explaining the mathematical background, it is important to mention about the matrix type often used in our research calculations. It is a matrix called column-stochastic which all the values greater than or equal to zero, and also the sum of the values of each column is equal to 1. If all the values are greater than zero, it is called a positive matrix with real numbers. Assume that we have a matrix "A" with transpose "A^t":

$$A = \begin{pmatrix} 1/2 & 0 & 1/3 \\ 0 & 1 & 1/3 \\ 1/2 & 0 & 1/3 \end{pmatrix} \quad \text{and} \quad A^t = \begin{pmatrix} 1/2 & 0 & 1/2 \\ 0 & 1 & 0 \\ 1/3 & 1/3 & 1/3 \end{pmatrix}$$

Figure 3.6 A matrix with transpose

As it is seen in Figure 3.6, all the values of the matrix “A” are positive, and the matrix “A” is a type of column-stochastic. But transpose “A^t” is a row-stochastic because the sum of the values of each row is equal to 1. As it is mentioned before, the matrix “A” and its transpose “A^t” have the same eigenvalue and it is equal to 1. Accordingly, we can say:

- Any column-stochastic matrix has an eigenvalue which is equal to 1.
- If our matrix is a column-stochastic matrix, the eigenvector corresponding to the eigenvalue (which is equal to 1) can have only positive values or only negative values.
- If our matrix is a positive column-stochastic matrix, there is a unique vector to the corresponding eigenvalue (which is equal to 1) and this unique vector has only positive values and the sum of the values is equal to 1 [60].

3.2.2 Directed Graphs

Graphs are objects reflecting the relationships between structures consisting of nodes or points. They are often used to represent the network formed by more than one web site on the internet [61].

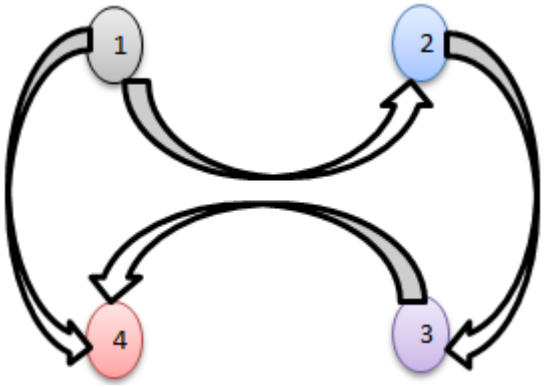


Figure 3.7 Directed graph with four nodes

As seen in Figure 3.7, we called “node” to each point and each node represents a web site on the internet graph. Each arrow between the nodes represents a link and it is also called “edge”. Therefore, this structure consisting of nodes and edges is called a graph. There are various graph types, but we focus on a particularly directed web graph in this study. Even if internet graphs can be very large, all graphs are thought as composed of finite points when calculations are done.

If there is an edge between a node “i” and a node “j” or vice versa, we can say that these nodes are adjacent in that directed graph, and node “i” and node “j” are the last points in the graph. If the edge between node “i” and node “j” exits from node “i” and enters to node “j”, the node “i” is called “tail” and the node “j” is called the “head” of the edge. For instance, as seen in Figure 3.7, there is an edge between node “1” and node “2”, so these nodes are adjacent and node “1” is the tail part, and node “2” is the “head” part. Since there isn’t any edge between node “1” and node “3”, we cannot say that they are adjacent nodes. The number of edges to a node is called inlinks (internal links) to a website, likewise, the number of edges from a node is called outlinks (external links) from a website.

When we look at the graph, node “1” has two outlinks and no inlinks. One of these outlinks goes to node “4” and the other one goes to node “2”. Likewise, node “2” and node “3” have just one outlink and node “4” has no outlink. But node “4” has two inlinks and one of these inlinks comes from node “1” and the other one comes from node “3”. Also, node “2” and node “3” have just one inlink.

We can make a few comments about our graph in Figure 3.7, in general. As it is known, if a website (a node) is offered (getting a link) from other websites, it means that that is a recommended website which is important. When we look at the graph, node “4” has more inlinks than others, so we can say that node “4” is the most important node in our graph. Likewise, node “1” has no inlinks, so we can say that node “1” is the least important node in our graph. Of course, all the comments we make are from what we see on the graph, but we know that there are other factors indicated the importance.

3.2.3 A glimpse of PageRank Computation

PageRank algorithm was the backbone of the Google search engine in the 2000s. The importance of a web page in the ranking pages result was determined by the PageRank algorithm when given an inquiry to the search engine. As is known, the basic logic of the PageRank algorithm is that the importance of a website increases so long as the number of inlinks increases. In this regard, the PageRank algorithm can be considered as an election logic. But in this algorithm one can distribute his/her vote between attendees. Namely, you can distribute your vote between other websites by giving links if you think those websites are the best to represent your website. Then, when given an

inquiry to the search engine, the algorithm controls the scores of each page in that related field. The website which has more inlinks than others comes to the top of the result list.

It is useful to explain one more thing in this election logic. Since the score of each website is calculated by inlinks, each website has a different score. So, it is important which website sent a link to your website and at what rate. Namely, because of getting more inlinks, bbc.com will have more effect than any ordinary website on your website if you get a link from. Of course, it is a dynamic structure and the score of each website is constantly changing. As a result, everyone's vote is not equal as in general election on the internet graph. Let's look at the mathematics of Google's PageRank.

Brin et al. summarized the PageRank calculation with a simple sum Formula (3.1) [62].

$$r(P_i) = \sum_{P_j \varepsilon B(P_i)} \frac{r(P_j)}{|P_j|} \tag{3.1}$$

As it is seen in Formula 3.1, PageRank of a page "i", $r(P_i)$, is calculated by collecting all PageRanks coming from other webpages to the page "i". $B_{(Pi)}$ represents the inlinks to the page P_i. $|(P_j)|$represents the outlinks from the page P_j. But how can we calculate $r(P_j)$? In order to overcome this problem, Brin and Page used an iterative calculation (3.2). According to this iterative calculation, each webpage has an equal PageRank score at the beginning of the calculation.

$$r_{k+1}(P_i) = \sum_{P_j \varepsilon B(P_i)} \frac{r_k(P_j)}{P_j} \tag{3.2}$$

The PageRank value of each P_i is calculated by getting one before the value of the P_j. So, in order to get $r_{k+1}(P_i)$ of page P_i at iteration k+1, we use the Formula 3.2. This process is started for all pages in the graph with $r_0(P_i) = 1/n$, where n is the number of pages in the related graph. To illustrate this calculation, let's apply on a simple graph.

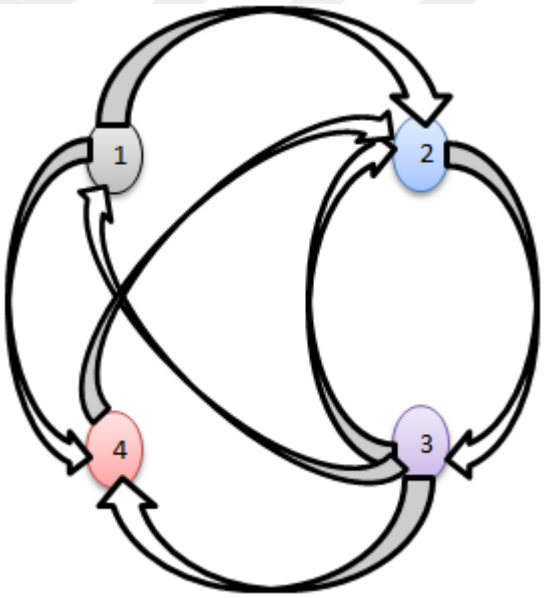


Figure 3.8 A graph with four nodes

As seen in Figure 3.8, we have a "1" point value and 4 pages. So, when we start to calculate, each node will get 1/4 scores at the beginning. In the first iteration, node 1 distributes its point between node 2 and node 4 (each node gets 1/8 points). Node 2 gives all its points to node 3. Node 3 distributes its point between nodes 1, 2, and 4 (each node gets 1/12 points). Lastly, node 4 gives all its points to node 2. At the end of the first iteration, node 1 has 1/12 points coming just from node 3. Node 2 has 11/24 (1/8+1/12+1/4) coming from node 1, node 3, and node 4 respectively. Node 3 has 1/4

points coming just from node 2. Lastly, node 4 has 5/24 (1/8+1/12) points coming from node 1 and node 3. Let's show the first two iterations in the table.

Table 3.1 First few iterates using on Figure 3.8

Iteration 0	Iteration 1	Iteration 2	Rank at iteration 2
$r_0(P_1) = 1/4$	$r_1(P_1) = 1/12$	$r_2(P_1) = 1/12$	4
$r_0(P_2) = 1/4$	$r_1(P_2) = 11/24$ → (1/8+1/12+1/4)	$r_2(P_2) = 8/24$ →(1/24+1/12+5/24)	2
$r_0(P_3) = 1/4$	$r_1(P_3) = ¼$	$r_2(P_3) = 11/24$	1
$r_0(P_4) = 1/4$	$r_1(P_4) = 5/24$ →(1/8+1/12)	$r_2(P_4) = 3/24$ →(1/24+1/12)	3

As seen at the end of the second iteration from Table 3.1, node 1 is the last page, and node 3 is the first page on the raking page result. Even if node 1 gets a link from the winner node 3, it couldn't pass the others. Because the winner node 3 distributes its point to all pages, so it is meaningless to get a link from node 3. Likewise, even if node 2 gets links from all nodes, node 3 comes first because of getting all points of node 2.

3.2.4 Matrix Representation to Compute Page Score

PageRank calculation can be considered as a matrix problem. We saw how we can calculate the PageRank score by the given formula (3.2). But it can be calculated easier and understandable way using matrices at each iteration. It can also be easier to apply other operations on a matrix. To accomplish this, we should just transform our graph to a

matrix structure. To illustrate, let's apply on a simple graph.

$$H_{ij} := \begin{cases} 1/N_i & \text{If there is a link from } P_i \text{ to } P_j \\ 0 & \text{Otherwise} \end{cases}$$

But first, we transform our graph into a matrix, for that we should obey the above rule. As can be seen in Figure 3.9, our directed graph consists of six nodes which represent a very small version of the web graph. There are two links from node 1 to node 2 and 3, respectively. It means that node 1 distributes its point between nodes 2 and 3 by half-and-half. Node 2 does not distribute its point. Node 3 distributes its point between nodes 1, 2 and 4 by a third, node 4 distributes its point between nodes 5 and 6 by half-and-

half, node 5 gives all its point to node 6, and lastly, node 6 distributes its point between node 4 and 5 by half-and-half. After distributing points, we can arrange our matrix:

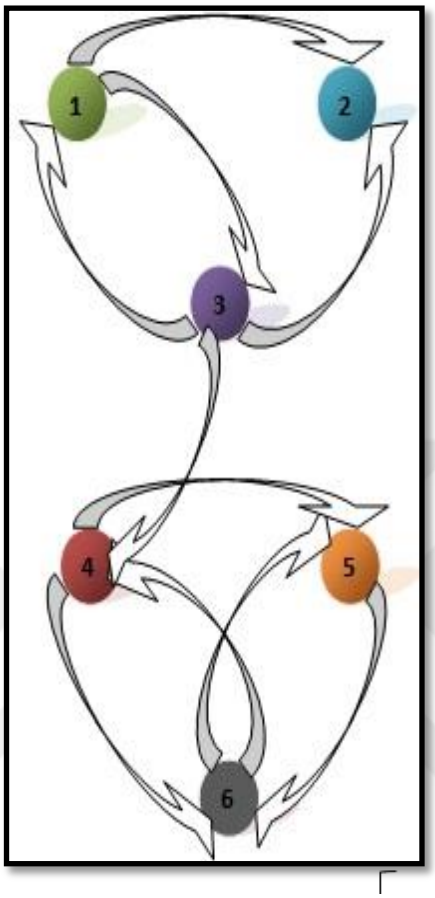


Figure 3.9 A graph with six nodes

$$H = \begin{bmatrix} 0 & 1/2 & 1/2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 1/3 & 1/3 & 0 & 1/3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1/2 & 1/2 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1/2 & 1/2 & 0 \end{bmatrix}$$

As it is seen in the "H" matrix, if there is no link from the page (node) P_i to another page, we put a "0" to that place. Namely,

we don't share P_i's point with that page. Accordingly, H_{ij} indicates a directed link from page "i" to page "j". Likewise, N_i indicates the total number of outlinks from page "i". Thus, each row represents the outlinks from page "i", whereas each column represents the inlinks to page "i". Then, we can calculate the PageRank score for each page according to the obtained values by the iterative formula:

$$r(P)^T_{(k+1)} = r(P)^T_{(k)} H, \qquad k = 0, 1, 2, \ldots \tag{3.3}$$

We denote the PageRank score with "π" in the following section. Thus,

$$\pi^T_{(k+1)} = \pi^T_{(k)} H, \qquad k = 0, 1, 2, \ldots \tag{3.4}$$

3.2.4.1 Random Walk on the Web Graph

It will be useful to know Random Walker in order to understand the structure and problems of the PageRank algorithm since it will be easy to comprehend the transition between pages by the Random Walk model. As the name implies, Random Walker starts to move by selecting a random page and move on to one another web page using one of the external links on this page. This move is repeated for each occurrence of a new web page. But there is one thing we should pay attention to this movement. If the Random Walker chooses web pages according to the external links, it means that when a web page has too much inlinks, the probability of the Random Walker chooses that page will be more than other pages. Another important point of the Random Walker movement is that the probability of a page being selected by the Random Walker is not relevant to the previous page [63]. Namely, assume that Random Walker passed from page "i" to page "j", the next movement of Random Walker is not affected by page "i". Random Walker goes on its way by choosing an outlink on the page "j". It means that the probability of a page being selected by the Random Walker is changed by the ratio of the distribution of page j's score to other pages. For more details you can read Markov chains in the section 3.2.5.

3.2.4.2 Dangling Nodes Problem in the PageRank Algorithm

We have seen how the Random Walker provides the transition between web pages. But if we look at the graph carefully in Figure 3.9, we can realize several problems when the Random Walker passes from one page to another. As we mentioned before, the Random

Walker passes to another page by choosing an outlink on the arrived page. But when the Random Walker arrives node 2, it cannot move from node 2 to another page since there are no outlinks on node 2. We called node 2 as a dangling node. For this reason, as it is seen in "H" matrix, all the entries are "0" on row 2. But how the Random Walker can move to another page in this circumstance. Actually, dangling nodes mostly consist of documents such as pictures, pdf, words, etc. in the gigantic web graph. Random Walker can stop the process, or it can start again from the beginning in such a case. But because of reducing the performance of the Random Walker, such a solution is not a very logical way. To overcome this problem, Brin and Page appealed to the following method [62].

If a webpage does not link to any other webpages, the PageRank score of that page will be distributed equally to all other webpages. For "n" dimensional matrices, all the entries of the row consisted of zeros will be replaced by 1/n. according to this process, our new matrix formulation will be as follows.

$$S = H + (1/n)\, de^{T} \tag{3.5}$$

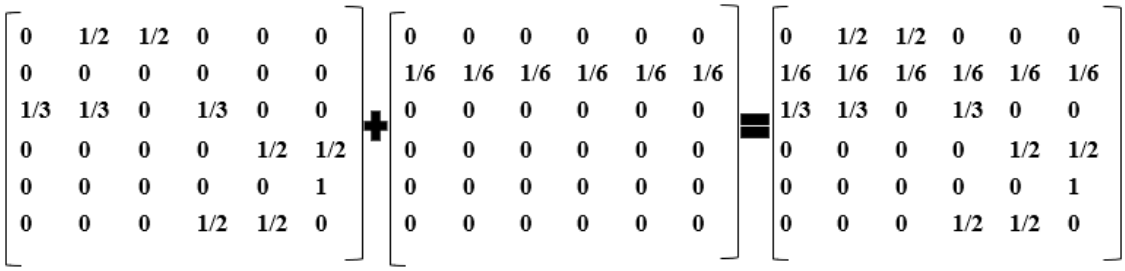
Where, "e" represents the column vector of all 1s and "d" indicates the dangling node and equal to "1", or "0" otherwise. It can be stated as below.

$$d_i := \begin{cases} 1 & \text{If page "i" is a dangling node} \\ 0 & \text{Otherwise} \end{cases} \qquad i = 1, 2, \ldots, n$$

If we apply this Formula (3.5) on our graph in Fig. 3.9, the d_2 column vector which is the second row consisted of zeros will get the value "1" in our matrix.

$$\mathbf{S = H + (1/n)de^{T} = H + (1/6)} \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \mathbf{(1\ \ 1\ \ 1\ \ 1\ \ 1\ \ 1)}$$

Accordingly, when we also add "H" matrix, S matrix will be as follows:



We get the above result. Namely, we used a row stochastic matrix in which the sum of all the entries is equal to 1 in order to get the "S" matrix. Even though we solved the dangling nodes problem in our graph, we have still some problems if we look carefully. Let's take a glance at the next problem.

3.2.4.3 Rank Sink Subgraphs Problem in the PageRank Algorithm

We have seen how to solve the dangling nodes problem in the PageRank algorithm. But when we take our graph as several subgraphs, we can see the Random Walker has another problem. Assume that Random Walker passed from node 3 to node 4, Random Walker will drive round and round in the subgraph consisted of nodes 4, 5, and 6 since there is no outlink from this subgraph to another consisted of nodes 1, 2, and 3. In this way, nodes 4, 5, and 6 will get more and more PageRank at each iteration. We called this problem as a "rank sink" which refuse to share PageRank via not giving a link to other nodes or subgraphs. We can also see this problem in Figure 3.7. In that directed graph node 4 is also a rank sink node. In order to overcome the problem that Random walker gets stuck in a subgraph, we transform our matrix into an irreducible matrix. Let's explain how to deal with this problem.

It is called the "teleportation" method providing the PageRank Algorithm to turn into an irreducible status. Even if there is a little chance, Random Walker will be able to make the transition between pages in this way. Let's represent this method with the Formula (3.6) below:

$$G = \alpha S + (1-\alpha)(1/n)ee^{T} \tag{3.6}$$

As in the previous formula, "e" represents the column vector of all 1s, "α" is the damping factor or breaking the power factor of the rank sink of subgraphs (teleportation probability factor) which is between "0" and "1" (generally it is equal to 0.85). Let's apply this formula to our matrix.

$$G = 0.85\begin{bmatrix} 0 & 1/2 & 1/2 & 0 & 0 & 0 \\ 1/6 & 1/6 & 1/6 & 1/6 & 1/6 & 1/6 \\ 1/3 & 1/3 & 0 & 1/3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1/2 & 1/2 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1/2 & 1/2 & 0 \end{bmatrix} + (1\text{-}0.85)\,(1/6)\begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}(1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1)$$

.

$$G = 17/20\begin{bmatrix} 0 & 1/2 & 1/2 & 0 & 0 & 0 \\ 1/6 & 1/6 & 1/6 & 1/6 & 1/6 & 1/6 \\ 1/3 & 1/3 & 0 & 1/3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1/2 & 1/2 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1/2 & 1/2 & 0 \end{bmatrix} + 1/40\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

.

$$G = \begin{bmatrix} 0 & 17/40 & 17/40 & 0 & 0 & 0 \\ 17/120 & 17/120 & 17/120 & 17/120 & 17/120 & 17/120 \\ 17/60 & 17/60 & 0 & 17/60 & 0 & 0 \\ 0 & 0 & 0 & 0 & 17/40 & 17/40 \\ 0 & 0 & 0 & 0 & 0 & 17/20 \\ 0 & 0 & 0 & 17/40 & 17/40 & 0 \end{bmatrix} + \begin{bmatrix} 1/40 & 1/40 & 1/40 & 1/40 & 1/40 & 1/40 \\ 1/40 & 1/40 & 1/40 & 1/40 & 1/40 & 1/40 \\ 1/40 & 1/40 & 1/40 & 1/40 & 1/40 & 1/40 \\ 1/40 & 1/40 & 1/40 & 1/40 & 1/40 & 1/40 \\ 1/40 & 1/40 & 1/40 & 1/40 & 1/40 & 1/40 \\ 1/40 & 1/40 & 1/40 & 1/40 & 1/40 & 1/40 \end{bmatrix}$$

.

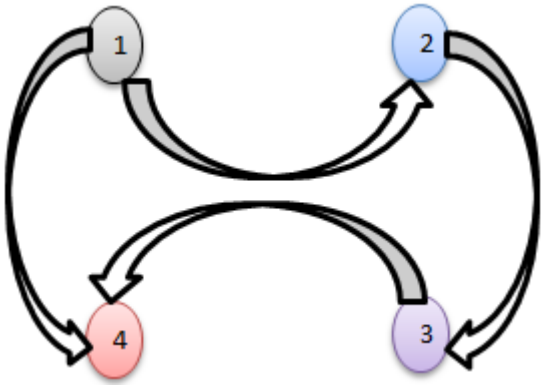
$$G = \begin{bmatrix} 1/40 & 9/20 & 9/20 & 1/40 & 1/40 & 1/40 \\ 1/6 & 1/6 & 1/6 & 1/6 & 1/6 & 1/6 \\ 37/120 & 37/120 & 1/40 & 37/120 & 1/40 & 1/40 \\ 1/40 & 1/40 & 1/40 & 1/40 & 9/20 & 9/20 \\ 1/40 & 1/40 & 1/40 & 1/40 & 1/40 & 7/8 \\ 1/40 & 1/40 & 1/40 & 9/20 & 9/20 & 1/40 \end{bmatrix}$$

In this way, we got the irreducible "G" matrix. Namely, we let the Random Walker able to make the transition between all pages in the graph by adding $(1-\alpha)(1/n)ee^T$ on our "S" formula.

3.2.4.4 Computation of the PageRank Vector

After solving the problems in the calculation of the PageRank Algorithm, it is time to calculate the PageRank vector which provides the importance of each page in order. For this purpose, let's review a few details of the calculation of the PageRank vector.

If the Random Walker can pass from node "i" to node "j" in a given graph, we can say that there is a path between node "i" and node "j" and it is called as a *connected graph*. But if the Random Walker has a chance to pass any node "i" to any node "j", we call these types of graphs as *strongly connected graphs*. As we saw in our graph, "H" matrix represents a connected graph (There is a link from node "1" to node "2" but there is no link from node "2" to any other page) but after solving dangling nodes and rank sinks problems, it is transformed into a strongly connected graph. In this way, Random Walker could make a transition between any web pages. Considering this case in terms of matrices, if $B = I + A + A^2 + A^3 + \ldots + A^k$ is a positive value, for a positive "k" value (multiplying our matrix by "k" times, i.e. A^k for matrix "A"), we can say that our matrix is a strongly connected graph. Here, adding the identity matrix is that the cycle of the nodes themselves. Accordingly, if there is a path from node "i" to node "j" at "k" times, it means that we can pass from node "i" to node "j". If the values of matrix "B" are all positive, it means we can pass from any node "i" to any node "j", in other words, it will be a strongly connected graph. To illustrate, let's look at our previous graph. Let's say that "1" for each passing from node "i" to node "j".



$$A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

Figure 3.10 Directed graph with four nodes

Accordingly, matrix "A" is a connected graph, not a strongly connected graph. Let's calculate matrix "B" according to the matrix "A".

$$B = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} + \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 2 & 1 & 1 \\ 1 & 1 & 0 & 1 \end{bmatrix}$$

As it is seen, since all the entries are not bigger than "1", matrix "B" is not positive. Namely, there is no connection from any node "i" to any node "j", matrix "B" is not a strongly connected graph.

Before computing the PageRank vector of matrix "G", let's take a glance at matrix "H" which is an untouched matrix before obtaining matrix "G". Therefore, we can compare our guesses are true or not at the end of getting the PageRank vector.

As seen in Figure 3.11, our graph consists of two subgraphs. The first subgraph consists of nodes 1, 2, and 3, the second subgraph consists of nodes 4, 5, and 6. When we look at the first subgraph, node 2 has two inlinks and no outlinks. So, node 2 most probably will be more important than others. Node 1 and node 3 get link from each other, but because of dividing its score into three parts, the link comes from node 3 is less important than 1. Therefore, node 3 will get more points than node 1. Consequently, our rank order will be like 2 > 3 > 1 in the first subgraph.

In the second subgraph, each node gets two inlinks, but since node 5 gives all its point to node 6, node 6 will be the most important node in the second graph. When we come to decide which one is more important between nodes 4 and 5, it is a bit complicated. Firstly, both node 4 and node 5 get a link from node 6. Thus, node 6 is not a determinant

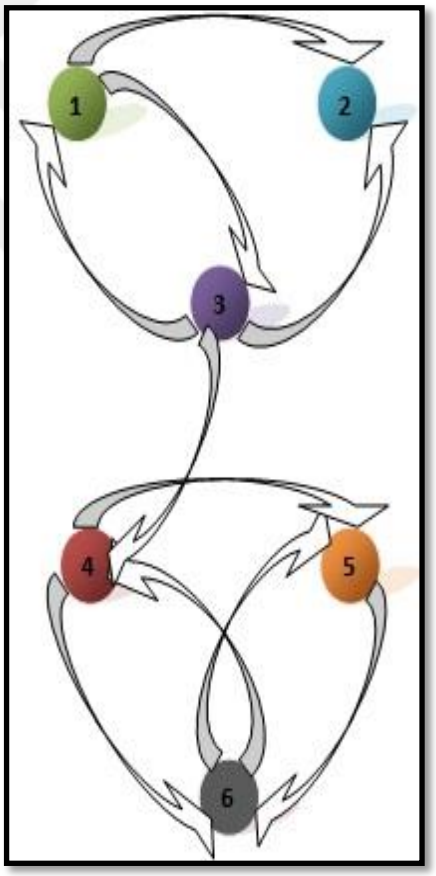


Figure 3.11 A graph with six nodes

node. In this case, if node 3 has more score than node 4, our rank order will be like 6 > 4 > 5, otherwise if node 4 has more score than node 3, our rank order will be like 6 > 5 > 4 in the second subgraph.

After guessing the PageRank score order, we can start to calculate our "G" matrix now. Due to the fact that all the entries in matrix "G" are bigger than "0", there is a path between any page "i" and "j". As it is done before in an example, we can start with dividing our PageRank (which is equal to 1) to each node in equal in order to start the iterative process. We have 6 nodes in our graph. It means each node will get a 1/6 PageRank score at the beginning of the iterative process. After the first process, the PageRank score of each node will change by the amount of importance value they have until they reach a threshold. After the threshold (after a certain iteration), each node will start to get an unchanged score, in other words, they will reach a balance or saturation point. This threshold (the number of iterations of the process) is changed according to the graph structure. We will get each result of the iterative process according to the logic as we mentioned before in Table 3.1. That operation is the same with multiplying matrix "G" with π^T (we denoted before with $r(P_i)$). Accordingly, our first step of importance vector will be as follows:

$$\pi^T_{(k+1)} = \pi^T_{(k)}\, G \tag{3.7}$$

According to Formula (3.7),

1.step → $\pi^T_{(1)} = \pi^T G$,

2.step → $\pi^T_{(2)} = (\pi^T G)G$,

3.step → $\pi^T_{(3)} = ((\pi^T G)G)G$

.

.

.

At the last step (threshold) we can see that the PageRank score of each node reaches a saturation point and after that saturation point (iteration), the results do not be changed. Let's calculate the first iterate of the PageRank vector calculation.

$$\pi^T = (1/6 \;\; 1/6 \;\; 1/6 \;\; 1/6 \;\; 1/6 \;\; 1/6) \quad G = \begin{bmatrix} 1/40 & 9/20 & 9/20 & 1/40 & 1/40 & 1/40 \\ 1/6 & 1/6 & 1/6 & 1/6 & 1/6 & 1/6 \\ 37/120 & 37/120 & 1/40 & 37/120 & 1/40 & 1/40 \\ 1/40 & 1/40 & 1/40 & 1/40 & 9/20 & 9/20 \\ 1/40 & 1/40 & 1/40 & 1/40 & 1/40 & 7/8 \\ 1/40 & 1/40 & 1/40 & 9/20 & 9/20 & 1/40 \end{bmatrix}$$

$$\pi^T_1 = \pi^T G = (0{,}095833333 \quad 0{,}166666667 \quad 0{,}119444444 \quad 0{,}166666667 \quad 0{,}190277778 \quad 0{,}261111111)$$

If we go on to compute the PageRank vector, we can get the ultimate PageRank vector (π^T_*). It is shown the PageRank vector of each iteration in Table 3.2 and the ultimate PageRank vector (π^T_*).

Table 3.2 Calculation of each page score at each iteration

π^T_2	0,082453704	0,12318287	0,089340278	0,193425926	0,230416667	0,281180556
π^T_3	0,067763985	0,102806809	0,077493731	0,187265721	0,244158661	0,320511092
π^T_4	0,061520855	0,090320549	0,068363992	0,197738069	0,255369444	0,326687092
π^T_5	0,057165209	0,083311572	0,063941774	0,196007223	0,260676104	0,338898118
π^T_6	0,054919309	0,079214523	0,061097686	0,198951009	0,264137242	0,341680231
π^T_7	0,053533069	0,076873775	0,059562764	0,198747167	0,265990334	0,345292892
π^T_8	0,052766568	0,075518122	0,058642006	0,199516047	0,267107476	0,346449781
π^T_9	0,052313636	0,074739427	0,058124192	0,199554793	0,267733878	0,347534075
π^T_{10}	0,052056607	0,074289902	0,057821381	0,199758589	0,268100854	0,347972668
π^T_{11}	0,051907127	0,074031185	0,05764846	0,199795511	0,268310187	0,348307529
π^T_{12}	0,051821482	0,073882011	0,05754828	0,199852182	0,268431543	0,348464502
π^T_{13}	0,051771964	0,073796094	0,057490748	0,199869378	0,268501209	0,348570607
π^T_{14}	0,051743492	0,073746577	0,057457531	0,199886	0,26854144	0,34862496
π^T_{15}	0,051727066	0,07371805	0,057438416	0,199892673	0,26856459	0,348659206
π^T_{16}	0,051717608	0,073701611	0,057427393	0,199897771	0,268577939	0,348677678
π^T_{17}	0,051712156	0,07369214	0,057421045	0,199900169	0,268585627	0,348688862
π^T_{18}	0,051709016	0,073686682	0,057417386	0,199901782	0,268590058	0,348695075
π^T_{19}	0,051707206	0,073683538	0,057415278	0,199902613	0,268592611	0,348698754
π^T_{20}	0,051706163	0,073681726	0,057414064	0,199903134	0,268594082	0,348700831
π^T_{21}	0,051705563	0,073680682	0,057413364	0,199903416	0,26859493	0,348702046
π^T_{22}	0,051705216	0,073680081	0,057412961	0,199903586	0,268595418	0,348702738
π^T_{23}	0,051705017	0,073679734	0,057412728	0,199903681	0,268595699	0,348703141
π^T_{24}	0,051704902	0,073679534	0,057412595	0,199903737	0,268595861	0,348703371
π^T_{25}	0,051704836	0,073679419	0,057412517	0,199903768	0,268595955	0,348703504
π^T_{26}	0,051704798	0,073679353	0,057412473	0,199903787	0,268596009	0,348703581
π^T_{27}	0,051704776	0,073679315	0,057412447	0,199903798	0,26859604	0,348703625

π^T_{28}	0,051704763	0,073679293	0,057412433	0,199903804	0,268596058	0,348703651
π^T_{29}	0,051704756	0,07367928	0,057412424	0,199903807	0,268596068	0,348703665
π^T_{30}	0,051704751	0,073679273	0,057412419	0,199903809	0,268596074	0,348703674
π^T_{31}	0,051704749	0,073679268	0,057412416	0,19990381	0,268596077	0,348703679
π^T_{32}	0,051704748	0,073679266	0,057412415	0,199903811	0,268596079	0,348703681
π^T_{33}	0,051704747	0,073679265	0,057412414	0,199903811	0,26859608	0,348703683
π^T_{34}	0,051704746	0,073679264	0,057412413	0,199903812	0,268596081	0,348703684
π^T_{35}	0,051704746	0,073679263	0,057412413	0,199903812	0,268596081	0,348703684
π^T_{36}	0,051704746	0,073679263	0,057412413	0,199903812	0,268596082	0,348703685
π^T_{37}	0,051704746	0,073679263	0,057412413	0,199903812	0,268596082	0,348703685
.	.	.	.	.	.	.
.	.	.	.	.	.	.
.	.	.	.	.	.	.
π^T_*	0,051704746	0,073679263	0,057412413	0,199903812	0,268596082	0,348703685

As it is seen our ultimate PageRank vector (π^T_*) fixated at iteration 36. As a result, our ultimate PageRank vector (π^T_*) is as follows:

π^T_* = 0,051704746 0,073679263 0,057412413 0,199903812 0,268596082 0,348703685

According to the π^T_*, our guess for the first graph is true and the order of the importance of the nodes is as 2 > 3 > 1. For the second subgraph, the order of the importance of the nodes is as 6 > 5 > 4 and the importance of node 4 is bigger than node 3 (4 > 3). Consequently, the order of the importance of the nodes is as 6 > 5 > 4 > 2 > 3 >1 in this tiny web. According to the result, the most important node is page 6, and the least important node is page 1. If we interpret the result, the Random Walker visit page 1 by 5.170% and page 6 by 34.870% of the time.

3.2.5 Markov Chains in PageRank Computation

Markov chains, which is a special kind of stochastic process is used in many engineering fields, especially in the search engines. In Markov chains, the probability of a case that will take place next is only affected by the current state. Namely, the case which will take place next is not affected by the situation that occurred before the current state. Let's explain it by the graph with six nodes in Figure 3.11. Assume that Random Walker passed from node 3 to node 4, the probability of Random Walker passes to any other nodes from node 4 is not affected by node 3. The probability of passing to any other nodes is determined by the distribution of node 4's score.

3.2.5.1 Graph Theory of Markov Chain

Markov chain is actually an application of graph theory. Graph theory, as we mentioned before, is a set of a graph in which nodes represent states and edges represent the transition between states. Markov chains determine a situation would change according to certain statistical values. But the realization of current changes is independent of past states. Therefore, when the current state is affected by the former state, the future states are affected by just the current state. In this context, if a state is only dependent on the former state, it is called a time-dependent Markov process. Based on statistical Markov model, we can show the probability of each stochastic event by the following formula:

$$P(x_{t+1} = x_{t+1} \mid x_t = x_t, x_{t-1} = x_{t-1}, \ldots, x_0 = x_0) = P(x_{t+1} = x_{t+1} \mid x_t = x_t) \quad (3.8)$$

In Markov chains, it is called the transition probability passing from the state "i" to subsequent state "j" at a time and depending on the time it can be expressed as follows:

$$P_{ij}^{t,t+1} = P(\pi(t+1) = j, \pi(t)=i) \quad (3.9)$$

Accordingly, in Markov chains, the conditional probability of occurrence from the state "i" to subsequent state "j" at a time is;

$$Pij = \frac{P(\text{the process of passing from state "i" to state "j"})}{P(\text{the process in state "i"})}, P = P_{ij}, \quad i = 1,2,\ldots N \quad j = 1,2,\ldots N \quad 3.10$$

We can show the conditional probability of occurrence as adjacent to Figure 3.12. N = 0 to reflect the current state, all states of the process are specified on each line for a given state. Accordingly, "i" indicates the number of row and π_i indicates a row vector in the transition matrix "P". For instance, π_3 represents the third row. So, the transition matrix "P" consists of π_i probabilistic vectors and each entry of π_i indicates the probability of passing from one state to another. If we denote this passing probability by "K", π_i consists of finite passing probabilities. Let's show this on the web graph:

P =

Present state	Next state					
	1	2	3	.	.	N
1	P_{ii}	.	.	.	.	P_{iN}
2	.	.	.	.	.	.
3	.	.	.	.	.	.
.	.	.	.	.	.	.
.	.	.	.	.	.	.
N	P_{iN}	.	.	.	.	P_{NN}

Figure 3.12 The conditional probability of occurrence

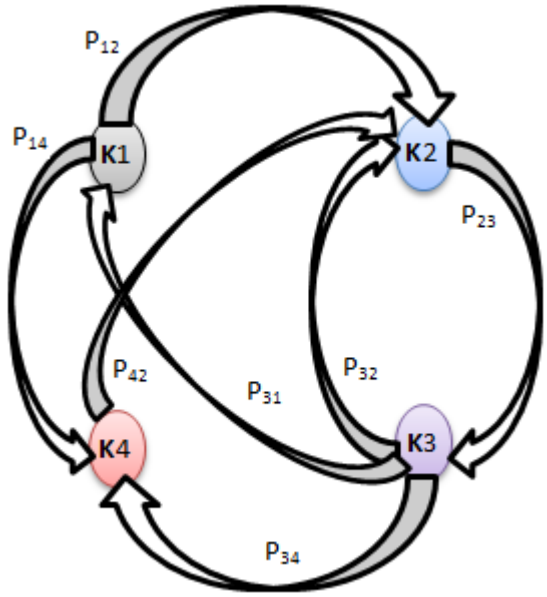


Figure 3.13 A four-node graph given with passing probability

Assume that N = 4 and we have four nodes K1, K2, K3, and K4. P_{ij} indicates the passing probability from one node to another. For instance, P_{23} indicates the passing probability starting from node K2 to node K3.

3.2.5.2 Formulizing Web Graphs with Markov Chains

Table 3.3 Passing probability of Random Walker between nodes

Present State (n=0)	Next State				
	K_1	K_2	K_3	K_4	(K_j)
K_1	0	1/2	0	1/2	
K_2	0	0	1	0	
K_3	1/3	1/3	0	1/3	
K_4	0	2	0	0	
(K_i)					

The Table 3.3 shows that the passing probability of Random Walker from node K_i to node K_j in the next step. Let's show these passing probabilities on the transition matrix.

$$\mathbf{P} = \begin{bmatrix} 0 & 1/2 & 0 & 1/2 \\ 0 & 0 & 1 & 0 \\ 1/3 & 1/3 & 0 & 1/3 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

$$0 \le P_{ij} \le 1$$

$$\sum_{j=1}^{4} P_{ij} = 1 \qquad (i = 1,2,3,4)$$

Our transition matrix satisfies the above conditions. Let's calculate the passing probability of Random Walker from node K_3 to node K_2 after two steps.

Table 3.4 Passing probability of Random Walker after two steps

Starting Node	First Step	Second Step	Probabilities	passing probability from node 2 to node 3
K_3	$K_1 = 1/3$	$K_1 = 0, K_2 = 1/2, K_3 = 0, K_4 = 1/2$	$K_3 \rightarrow K_1 \rightarrow K_2 = 1/3 * 1/2 = 1/6$	**1/6+1/3 =1/2**
	$K_2 = 1/3$	$K_1 = 0, K_2 = 0, K_3 = 1, K_4 = 0$	$K_3 \rightarrow K_2 \rightarrow K_2 = 1/3 * 0 = 0$	
	$K_3 = 0$	-	0	
	$K_4 = 1/3$	$K_1 = 0, K_2 = 1, K_3 = 0, K_4 = 0$	$K_3 \rightarrow K_4 \rightarrow K_2 = 1/3 * 1 = 1/3$	

As it is seen in Table 3.4, the passing probability of Random Walker from node K_3 to node K_2 after two steps is 1/2. Therefore, we can formulate this movement as follows:

$$\pi_i^{n+1} = \pi_i^n \text{ x } P \tag{3.11}$$

"π" is the probability vector. So, for n = 2;

$$\pi_3^2 = \pi_3^1 \text{ x } P = (1/3 \quad 1/3 \quad 0 \quad 1/3) \text{ x } \begin{bmatrix} 0 & 1/2 & 0 & 1/2 \\ 0 & 0 & 1 & 0 \\ 1/3 & 1/3 & 0 & 1/3 \\ 0 & 1 & 0 & 0 \end{bmatrix} = (0 \quad 1/2 \quad 1/3 \quad 1/6)$$

Thus, Random walker would be at K_1 with probability 0, at K_2 with probability 1/2, at K_3 with probability 1/3, and at K_4 with probability 1/6. We can get all passing probabilities of Random Walker by multiplying matrix "P" for the two steps.

$$P^2 = \begin{bmatrix} 0 & 1/2 & 0 & 1/2 \\ 0 & 0 & 1 & 0 \\ 1/3 & 1/3 & 0 & 1/3 \\ 0 & 1 & 0 & 0 \end{bmatrix} X \begin{bmatrix} 0 & 1/2 & 0 & 1/2 \\ 0 & 0 & 1 & 0 \\ 1/3 & 1/3 & 0 & 1/3 \\ 0 & 1 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 1/2 & 1/2 & 0 \\ 1/3 & 1/3 & 0 & 1/3 \\ 0 & 1/2 & 1/3 & 1/6 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

According to the result, some entries still are zero. It means that Random Walker can't pass some nodes. In order to overcome this problem, we mentioned some solutions such

as dangling node problems and rank sinks problems. As a result, time-dependent Markov chains are perfect to fit for some search engines.

3.3 Dataset

Table 3.5 Rating Dataset

Order	User_id	Item_id	ratings
0	1	1	3
1	1	2	4
2	1	3	4
3	1	4	5

For the research, we got a real-world dataset from http://www.jiliang.xyz/trust.html or https://www.cse.msu.edu/~tangjili/datasetcode/truststudy.htm because of having a trust relationship between users. It is publicly available and widely used to evaluate recommendation systems in the literature. The dataset consists of two sub-data sets. The first one is the rating dataset indicating "user id", "item id", "ratings" given to the related product by the related user and contains star values ranging from 1 to 5. The value of "1" indicates that the user does not like the item at all, while the value of "5" indicates that he likes it very much. If the ratings column contains a value of "0", it indicates that the relevant user did not score the relevant product. The ratings column consists of 48.891% 5 star, 31.152% 4 star, 11.016% 3 star, 5.04% 2 star, 3.889% 1 star, respectively. There are unique 7.353 users and 105.582 items in the dataset. So, when we look at Table 3.5 user 1 gives to item 1 "3" stars. And our dataset consists of 284086 rows with 3 columns. It is a ".csv" file and 4.82 Mb.

Table 3.6 Trust Network Dataset

	Trustee	trustor
0	1	3
1	1	4
2	1	5
3	1	6

The second dataset shows the trust relationship between users. The trust-network dataset indicates the trustee and the trustor respectively. So, "user 1" trust to "user 3", "user 4", "user 5", and so on. And our trust network dataset consists of 111781 rows with 2

columns. It is a ".csv" file and 998 Kb. Our algorithms are executed on Jupyter Notebook with python version "2.7.11".

3.4 Recommender Model Based on Trust Relationship

We use the PageRank graph theory in order to find the reliable or trusted users in our dataset. In our recommendation model, we consider a user as a website or a node in a graph. So, we have a set of users U = {u_1, u_2, …., u_m} and a set of items I = { i_1, i_2, …., i_n} and every user have a set of rates Ru_i = { u_{i1}, u_{i2}, …., u_{im}} and it is represented by $r_{u,i}$ for user "u" on item "i". Lastly, we also have a trust network among users. If user "u" trusts user "v", then we represent by $t_{u,v}$ for the value of this trust with a real number between "0" and "1" and "0" means no trust and "1" means full trust.

3.4.1 Creating User Matrix for Creating the Trust Relationship

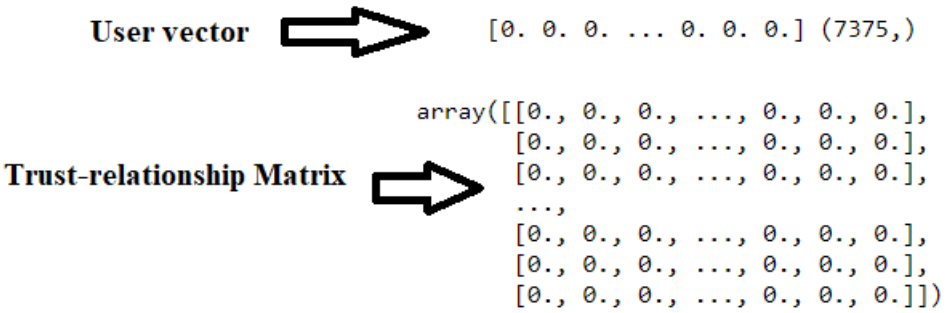


Figure 3.14 User vector with Trust-relationship Matrix

As you can see in Figure 3.14 from the output of the program, we have a total of 7375 users, and we define a vector for these users. Then, we define the matrix for showing each user's trust relationship with other users.

3.4.2 Specify Relationship Between Users

```
array([0., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
       1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
       1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
       1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
       1., 1., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.])
```

Figure 3.15 The appearance of first user's vector in trust-relationship matrix

As you see in Figure 3.15 from the output of the program, if the user trusts the other user we placed “1” in the relevant cell, “0” if he/she does not. The vector in the Figure 3.15 shows just the trust relationship between the first user and just the first 200 other users.

3.4.3 Sharing the Trust Values Between Trustee of the Trustor “H” Matrix

```
[0.         0.01449275 0.01449275 0.01449275 0.01449275 0.01449275
 0.01449275 0.01449275 0.01449275 0.01449275 0.01449275 0.01449275
 0.01449275 0.01449275 0.01449275 0.01449275 0.01449275 0.01449275
 0.01449275 0.01449275 0.01449275 0.01449275 0.01449275 0.01449275
 0.01449275 0.01449275 0.01449275 0.01449275 0.01449275 0.01449275
 0.01449275 0.01449275 0.01449275 0.01449275 0.01449275 0.01449275
 0.01449275 0.01449275 0.01449275 0.01449275 0.01449275 0.01449275
 0.01449275 0.01449275 0.01449275 0.01449275 0.01449275 0.01449275
 0.01449275 0.01449275 0.01449275 0.01449275 0.01449275 0.01449275
 0.01449275 0.01449275 0.01449275 0.01449275 0.01449275 0.01449275
 0.01449275 0.01449275 0.01449275 0.01449275 0.01449275 0.01449275
 0.01449275 0.01449275 0.01449275 0.01449275 0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.        ]
```

Figure 3.16 Sharing the trust value of user 1 between trustee

As mentioned earlier, we share the trust score of the relevant user equally with other trusted users. We called the resulting matrix as the "H" matrix. What you see in Figure 3.16, it is the amount of trust that the first user shares with the other users he trusts.

Again, the vector in Figure 3.16 shows just the trust values between the first user and just the first 100 other users.

3.4.4 Solving Dangling Nodes Problem "S" Matrix

As you can see in Table 3.7, each user has a different number of trusted users, and each user's trust value is shared among other users he or she trusts. However, sometimes some users do not have a trust relationship with other users in the dataset. Users in this state are called "Dangling Users". Because these users have no trust sharing, they cause problems with matrix multiplications. Therefore, their trust values are shared equally among all other users. It means that if you don't trust anyone, you trust everyone. As seen in Table 3.7, user "155" doesn't trust anyone, while user "158" trusts 50 other users in the dataset.

Table 3.7 Dangling Users

User Order	Number of the Trustee
152	22
153	10
154	25
155	0
156	43
157	11
158	50

```
[0.00013559 0.00013559 0.00013559 0.00013559 0.00013559 0.00013559
 0.00013559 0.00013559 0.00013559 0.00013559 0.00013559 0.00013559
 0.00013559 0.00013559 0.00013559 0.00013559 0.00013559 0.00013559
 0.00013559 0.00013559 0.00013559 0.00013559 0.00013559 0.00013559
 0.00013559 0.00013559 0.00013559 0.00013559 0.00013559 0.00013559
 0.00013559 0.00013559 0.00013559 0.00013559 0.00013559 0.00013559
 0.00013559 0.00013559 0.00013559 0.00013559 0.00013559 0.00013559
 0.00013559 0.00013559 0.00013559 0.00013559 0.00013559 0.00013559
 0.00013559 0.00013559 0.00013559 0.00013559 0.00013559 0.00013559
 0.00013559 0.00013559 0.00013559 0.00013559 0.00013559 0.00013559
 0.00013559 0.00013559 0.00013559 0.00013559 0.00013559 0.00013559
 0.00013559 0.00013559 0.00013559 0.00013559 0.00013559 0.00013559
 0.00013559 0.00013559 0.00013559 0.00013559 0.00013559 0.00013559
 0.00013559 0.00013559 0.00013559 0.00013559 0.00013559 0.00013559
 0.00013559 0.00013559 0.00013559 0.00013559 0.00013559 0.00013559
 0.00013559 0.00013559 0.00013559 0.00013559 0.00013559 0.00013559
 0.00013559 0.00013559 0.00013559 0.00013559]
```

Figure 3.17 Sharing trust value of user 155 between all users

As you can see from the program output in Figure 3.17, all trust value of user 155 is shared equally among other users since he does not trust anyone. As we mentioned before, the matrix formed after this process is called the "S" matrix. Again, the vector in

Figure 3.17 shows just the trust values between the first user and just the first 100 other users.

3.4.5 Solving Rank Sink Subgraphs Problem

As we mentioned before, in order to overcome the problem that Random walker gets stuck in a subgraph and to break the power of the subgraphs, we transform our matrix into an irreducible matrix.

For this purpose, we use the teleportation method to give a little chance to Random

```
[[  2.03389831e-05   2.03389831e-05   2.03389831e-05 ...,   2.03389831e-05
    2.03389831e-05   2.03389831e-05]
 [  2.03389831e-05   2.03389831e-05   2.03389831e-05 ...,   2.03389831e-05
    2.03389831e-05   2.03389831e-05]
 [  2.03389831e-05   2.03389831e-05   2.03389831e-05 ...,   2.03389831e-05
    2.03389831e-05   2.03389831e-05]
 ...,
 [  2.03389831e-05   2.03389831e-05   2.03389831e-05 ...,   2.03389831e-05
    2.03389831e-05   2.03389831e-05]
 [  2.03389831e-05   2.03389831e-05   2.03389831e-05 ...,   2.03389831e-05
    2.03389831e-05   2.03389831e-05]
 [  2.03389831e-05   2.03389831e-05   2.03389831e-05 ...,   2.03389831e-05
    2.03389831e-05   2.03389831e-05]]
```

Figure 3.18 Rank sink part in G formula

$$G = \alpha S + \underbrace{(1-\alpha)(1/n)ee^{T}}_{\text{Rank Sink Part}}$$

Walker to able to make a transition between users.

In our calculations, we have given a value of 0.85 to the damping factor "α". This value is a standard accepted in many studies. Therefore, Random Walker can move to any users or subgraphs at a rate of 0.15 as we showed in the rank sink part in Formula (3.6). As you see in Figure 3.18, all users have the same chance of Random Walker being able to stop by themselves in the rank sink part.

3.4.6 Computation of "G" Matrix

```
[[2.03389831e-05 1.23391796e-02 1.23391796e-02 ... 2.03389831e-05
  2.03389831e-05 2.03389831e-05]
 [1.70020339e-01 2.03389831e-05 2.03389831e-05 ... 2.03389831e-05
  2.03389831e-05 2.03389831e-05]
 [2.03389831e-05 3.15018205e-02 2.03389831e-05 ... 2.03389831e-05
  2.03389831e-05 2.03389831e-05]
 ...
 [2.03389831e-05 2.03389831e-05 2.03389831e-05 ... 2.03389831e-05
  4.25020339e-01 2.03389831e-05]
 [2.03389831e-05 2.03389831e-05 2.03389831e-05 ... 4.25020339e-01
  2.03389831e-05 2.03389831e-05]
 [2.03389831e-05 2.03389831e-05 2.03389831e-05 ... 2.03389831e-05
  2.03389831e-05 2.03389831e-05]]
```

Figure 3.19 "G" Matrix after solution of rank sink problem

As shown in Figure 3.19, after the solution of the rank sink problem, we have obtained the "G" matrix. Therefore, the "G" matrix becomes an irreducible matrix. Namely, Random Walker can move between all users in our "G" matrix.

3.4.7 Computation of the PageRank Vector:

```
[[ 0.00013559  0.00013559  0.00013559 ...,  0.00013559  0.00013559 0.00013559]]
```

As we explained before, we start with dividing our PageRank (which is equal to 1) to each user in equal in order to start the iterative process. We have 7375 users in our graph. It means that each user gets 1/7375 trustworthiness score at the beginning of the iterative process as you see above. Let's go on the iterative process until the threshold. As we mentioned before, after a certain iteration each user will get an unchanged score.

```
[[  7.62288666e-04   1.35979139e-04   2.70471739e-04 ...,6.16892858e-05  6.16892858e-05   3.54713391e-05]]
```

After "59" iterations, we get the ultimate PageRank vector (π^{T}_{*}), and as you see above each user gets his/her own score. Let's look at a few users who get the most trust value.

Table 3.8 Trustworthiness of users in order

Order	User	Trustworthiness value
1	260	0.00150946708019796
2	5957	0.0010692603167929203
3	536	0.001049247566604396
4	3555	0.0010418192034589357
5	3556	0.0010405252123860767

As seen in Table 3.8, when we executed our algorithm on our dataset as we mentioned above, we got the trustworthiness score of each user in order. For instance, user 260 has approximately a "0,00150" point whereas user 5957 has almost a "0,00107" point. Namely, we can say that user 260 is more trusted than user 5957. So now we can calculate ratings of items based on trustworthiness score. Generally, any item's rating is calculated by the average of all users' ratings who has a preference/rate that item before but now we can calculate by looking at the trustworthiness score of user's. In other words, we calculate the rating of an item with a weighted average by trustworthiness score. In short, if user 1 has more trustworthiness score than user 2, it will affect the rating of that item more weighted by his/her trustworthiness score.

3.4.8 Findings after Computation of the PageRank Vector

Table 3.9 Comparing Average Rating Score and Weighted Rating Score Based on Trustworthiness

Item_id	Number of Users	Average Rating Score	Rating Score Based on Trustworthiness
1	1	3	3
4	3	3,666	3,986
17	5	4,8	4,866
31	5	4,4	4,340
33	12	3,91	4,155
35	8	4,15	3,81
491	972	4.201	4.197
577	316	2.471	2.402
612	119	4.050	4.045
645	187	4.074	4.077

$$\text{Weighted Average} = (\sum X_i W_i)/W_i \ = (\Sigma \text{ Rating*Trustfulness}) / \text{Trustfulness} \quad (3.11)$$

As it is seen in Table 3.9, for item 1, there is no difference between Average Rating Score (ARS) and Weighted Rating Score Based on Trustworthiness (WRSBT) both are "3.0". But when we look at item 4, 3 users rated this item, and the difference is almost 0.3194. This means that some of these users have more trustworthiness value and rated

this item more than average. Let us look at who rated this item “4” and what are their trustworthiness score.

Table 3.10 Trustworthiness of the User who rated item 4

User_id	User’s rating for item 4	Trustworthiness of the user
1	5	0,000763
83	3	0,000307
244	3	0,000114

As you can see in Table 3.10, user 1, user 83, and user 244 rated item 4. Since the user 1 is more a trustful user and rated with 5 points to item 4, WRSBT of item 4 is bigger tha n ARS.

The other important result is the average distance ratio between ARS and WRSBT when the number of users who rated the related items. Let us see the average difference between ARS and WRSBT when the number of rated users increases on 10.000 items in our dataset.

Table 3.11 Changing the distance between ARS and WRSBT by the different range of users

Number of users	**2-5**	**6-10**	**11-20**	**21-50**	**21-50**	**> 100**
Number of items	3521	1104	753	403	73	51
Average Difference	0.2368	0.1888	0.1484	0.1209	0.0901	0.0408

According to Table 3.11, 3521 items were rated by between 2 and 5 users and the average distance between ARS and WRSBT is 0.2368, likewise, 51 items were rated by more than 100 people and the average distance between ARS and WRSBT is 0.0408. Moreover, 94.73% of items were rated by less than 20 people. It means that it is easy to change the rating of an item by fake accounts.

3.4.9 Calculation of Items’ Ratings Based on Trusted Users via Iem-based Recommendation

We calculate the rating scores of products according to the trustworthiness value of each user who rated relevant products. In this part, we try to predict the rating score of products based on trusted users especially for the missing values. To accomplish this goal, after calculating the trust values of each user and finding the most trusted users as it is seen in Table 3.8, we get our rating dataset to train our model with 7375 users and

105114 items. We train the model with ranking factorization recommender for recommendations and get 0.384422 final training RMSE.

Table 3.12 ARS and WRSBT by first the 500 trusted users

Items	ARS	WRSBT	ARS by first 500 trusted users	WRSBT by first 500 trusted users
4	3,666	3,986	2,749	2,730
17	4,8	4,866	3,026	3,067
31	4,4	4,340	3,097	3,077
33	3,91	4,155	3,073	3,102
35	4,15	3,81	3,222	3,304
491	4.201	4.197	4,019	4,160
577	2.471	2.402	2,966	2,829
612	4.050	4.045	3,721	3,829
645	4.074	4.077	3,596	3,799

For getting a prediction by the first 500 trusted users (this number can be changed according to the used e-commerce platform), we use item-based recommendation. Therefore, we calculate a prediction for each item by looking at each user's experiences via item-based recommendation. As seen in Table 3.12, we calculate ARS and WRSBT for a couple of items by the first 500 trusted users. As we know that in our dataset 94.73% of items were rated by less than 20 people, So, it is important to consult experienced or trusted users to get opinions about related items especially for some e-commerce platforms like websites selling electronic products. According to the results in Table 3.12, the product rating score of the items rated by a few users (items 4, 17, 31, 33, 35) decreases significantly at different rates by ARS by the first 500 users. This means that, according to the first 500 trusted users, these items are not as good or valuable as it is thought. But for the items rated by many users (items 491, 577, 612, 645), the difference between ARS/WRSBT and ARS/WRSBT by the trusted users decreases.

The next calculation is the weighted rating score based on trustworthiness (WRSBT), this time we calculate the rating score of the products based on trust values of the first 500 trusted users. According to the results in Table 3.12, each rating score of the products varies slightly between ARS by the first 500 trusted users and WRSBT by the first 500 trusted users. But again, the product rating score of the items rated by a few

users (items 4, 17, 31, 33, 35) decreases significantly at different rates by ARS by the first 500 users.

We know that when a product is rated by a large number of users, the value it deserves emerges and the effects of the fake accounts on the results decrease. Therefore, the decrease in the difference between ARS/WRSBT and ARS/WRSBT by the trusted users for the items 491, 577, 612, 645 shows that we made an accurate calculation with our model that calculating the product rating score based on trusted users. We got the same results by executing the program many times for all the other items as well.

Apart from Ranking Factorization, different similarity measures can be used to find similarities between products in the item-based model, as we mentioned in the previous chapter. Table 3.13 shows the RMSE results of some similarity measures in the scikit learn library, which gives the best result on our own data set.

Table 3.13 Comparing Similarity Measures on Items

	Similarity Measure	**RMSE**
1	correlation	0.23597
2	braycurtis	0.23730
3	hamming	0.24641
4	cosine	0.25982
5	kulsinki	0.27004
6	rogerstanimoto	0.27004
7	jaccard	0.27005
8	dice	0.27005
9	matching	0.27005
10	canberra	0.35545

3.4.10 Calculation of Items' Ratings Based on Trusted Users via User-based Recommendation

In this part, we predict the rating score of the products based on trusted users via user-based recommendation. To accomplish this goal, we use a similar formula called TrustWalker which combines trust-based and item-based recommendation from Deng, Huang, and Xu [7].

Therefore, the task of our recommender algorithm is as follows. Given an item $i \in I$ for which $r_{tu,i}$ is unknown, predict the rating by trusted users on item i. we call "tu" trusted users and "i" for the target item. The predicted rating is represented by $\hat{r}_{tu,i}$.

Generally, traditional recommender algorithms predict $\hat{r}_i$ based on the average of all given ratings to item "i". Basically, algorithms aggregate all ratings given to the target item and calculate the average of the ratings. However, in trust user-based recommendation, the trust relationship between users is used instead of the average rating. In order to predict a rating of a product, we ask directly trusted users whether they know the rating for the target item especially for the missing values or items which are rated by a few users. If so, the algorithm returns with that value, otherwise, they recursively consult users whom they trust. The trusted users in our recommendation model are defined on the row of source trusted users in the transition matrix. Consequently, the rating of the target item is the aggregation of all ratings based on the trusted users' ratings returned by different random walks. And a single random walk is as follow:

Starting from the most trustful user on clustered domain, the random walker tries to obtain the rating of the target item given by the visited trusted user.

-If the visited trusted user has the rating on the target item, the random walker returns with that rating value and it stops walking. If the visited trusted user doesn't have a rating on the target item, then;

-Random walker jumps to another user who is trusted by the target trusted user via user-based recommender.

-If the random walker cannot find the rating of the target item on the visited trusted user, it will continue forever. To overcome this problem, we terminate the walk when the random walker goes very far away from the first visited trusted user. Because when the random walker goes far away from the first visited trusted user, the rating captured by the random walker will be less trustworthy. Because of this reason we define the maximum depth (the number of iteration) to "6" as similar [7]. If the random walker cannot find the rating value of the target item at iteration six, we get the rating calculating the most similar item on the first trusted user via item-based recommender.

Table 3.14 WRSBT by the first 10 trusted users via user-based recommender

Items	ARS	WRSBT	WRSBT by the first 10 trusted users via user-based recommender
4	3,666	3,986	2,997
17	4,8	4,866	3,233
31	4,4	4,340	3,379
33	3,91	4,155	3,045
35	4,15	3,81	3,291
491	4.201	4.197	4,338
577	2.471	2.402	2,588
612	4.050	4.045	4,120
645	4.074	4.077	4,130

As seen in Table 3.14, we calculate WRSBT by the first 10 trusted users (this number can be changed according to the used e-commerce platform) via user-based recommender. According to the results, we get almost the same results calculated by the item-based recommender. Namely, the rating score of the items rated by a few users decreases significantly at different rates but the rating score of the items rated by a large number of users varies slightly between ARS/WRSBT and WRSBT via user-based recommender. But if we compare item-based and user-based recommender to each other, user-based recommender's results are closer to ARS/WRSBT as we observe by experimental results executing many times.

Apart from Ranking Factorization, different similarity measures can be used to find similarities between users in the user-based model, as we mentioned in the previous chapter. Table 3.15 shows the RMSE results of some similarity measures in the scikit learn library, which gives the best result on our own data set.

Table 3.15 Comparing Similarity Measures on Users

	Similarity Measure	RMSE
1	jaccard	0.20361
2	rogerstanimoto	0.20361
3	dice	0.20361
4	matching	0.20361
5	kulsinki	0.20361

6	hamming	0.20364
7	cosine	0.20385
8	correlation	0.20972
9	braycurtis	0.21104
10	canberra	0.30736

3.5 Conclusion

In this chapter, we tried to show how fake accounts affect rating scores of items in e-commerce platforms and how to overcome these types of problems. For this purpose, we analyzed the relationship between users, and we found a trustworthiness value for each of them. Thus, we calculated the rating score of each item by the weighted average of users' ratings according to their trustworthiness values instead of getting a direct average of users' ratings. According to the calculations, the items rated by between 2 and 20 people have a great difference between ARS and WRSBT. It means that when the number of users who rated the item decreases, the effect of the fake account goes up. On the other hand, when the number of users increases, especially more than 100 people, the difference between ARS and WRSBT decreases almost "0". Besides this, we calculated the rating score of the products based on item-based recommender by the first 500 most trusted users and user-based recommender by the first 10 most trusted users. Similar to one before calculation, the rating score of the products changes significantly at different rates for the items rated by a few users but for the items rated by many users, results are close to results of the item-based and user-based recommenders. Actually, this is also proof that our model works very well. Consequently, if we think databases which are suffering from sparsity problems, this model can be a nice solution. By this model, items may get deserved rating scores more than in the traditional models.

4. A RECOMMENDER MODEL BASED ON TIME DECAY

Most of the existing product rating score algorithms ignore the time decay of users' ratings when creating a list of recommendations. The time decay of users' ratings to an item may improve the quality of product rating scores in e-commerce platforms, especially when it is thought that the majority of customers read the reviews before making a purchase.

In this chapter, we first introduce the concept of time decay by explaining its mathematical definition and redefine the product rating score based on time decay of the users' ratings. Besides, we calculate the product rating score based on the trust value of users by looking at the trust relationship. After that, we execute both algorithms together in order to show their both effects on the quality of the product rating score. Finally, we present experimentally the effectiveness of three approaches on a large real dataset.

Online consumer reviews bring a number of benefits such as saving time and money, finding experienced products by other consumers. But most of the e-commerce platforms do not have any verification or authentication mechanisms on their online users' reviews related to their products even if online ratings and reviews have become quite determinant on customers' purchasing decisions. Report [64] indicates that almost 82% of internet customers read reviews before making a purchase. This information shows us that the importance of online reviews/ratings has a great effect on the purchasing behavior of customers in e-commerce platforms. Even if 78% of customers think the information found online is vital and more trustful than advertisements, it is confirmed that most of the reviews are fake [65]. Therefore, time is also an important factor while calculating product rating scores especially in some e-commerce platforms such as hotels, restaurants, travel agencies, and other service-based companies. Most of the existing algorithms calculate rating scores of products or services based on average ratings but they ignore the time of each user's rating. As we mentioned above, for some e-commerce platforms rating time is a crucial factor since their products or services change over time. That is to say, a rating given yesterday, and a rating given ten years ago should not be considered as having the same value when calculating a hotel's rating

score. For this purpose, we propose a simple algorithm considering rating time when calculating a product or a service rating score. At the end of this chapter, for the calculation of a product rating score, our algorithm finds each user's trust value based on the PageRank algorithm by looking at given users' relationships as we mentioned in the previous chapter, and at the same time, it calculates the time decay of users' ratings to that product. Thus, each user affects the rating score of the products regarding his/her trust value and time decay of his/her rating.

4.1 Related Work

The latest researches show that the time factor significantly increases the quality of recommender systems, especially on e-commerce platforms. According to the observations, more recent reviews and ratings better reflect the quality of products and services. In this section, we review several major approaches for the time-based recommendations in general.

Lee et al. [66] present a novel based approach to building a recommender system based on implicit feedback. According to the empirical results, using two kinds of temporal information such as user rating time and product launching time improves both recommendation accuracy and performance.

Jamali and Ester [23] investigate whether a user rates after being exposed to an item rated by the target user's neighbors at a certain time. According to the experiment results, the influence of direct neighbors or rating items in the social network is higher than in the similarity network on datasets when the user is exposed to an item at a time.

Zheng and Li [24] propose a new computational approach using tag and time information. For this purpose, the authors use three strategies which are "tag weight", "time weight" and "tag and time". They use the "tag and time" strategy for calculating the target user's rating values with the combination of tag and time information. According to the experimental results, these three strategies give good results to personalize navigational recommendation rather than the traditional log-based method.

Raju et al. [67] propose an approach using a graph-based structure that uses the relationships between customers, products, customers and products. The authors utilize a matrix that consists of visiting area information, visiting date, visiting time, need type and satisfaction level for recommendations. They apply Collaborative Filtering to find

the most similar users based on filtered items and other user's additional information including day, time and need type, etc.

Ullah, Sarwar, and Lee [28] offer an interesting study that is the use of the recommender systems in a different area. The authors propose a smart device that recommends TV programs according to user preferences and the user's social network data. To calculate the rating value, they divide the time which the target user spends on the program during the broadcast, with the total time of the program.

Celdrán et al. [36] present a hybrid recommender and to compute the users' tracking, the authors calculate the number of times of visits and direction of the user to recommend items on that location, and lastly the date when the target user visited the related item last time.

Yang et al. [37] present a hybrid recommender model that considers the time of the target user's interest. According to the authors, the performance of the recommender system is changed as regards the time of the user selecting an item. Namely, the current interest of a user is more effective on the performance of the recommender system.

Zhang et al. [68] present a time series analysis for dynamic-aware recommendations to overcome data insufficiency. The developed algorithm called FARIMA deals with the year-long period of purchasing data to provide daily aware predictions.

Jiang et al. [69] propose an algorithm that uses time decay to provide dynamic item-based top-*N* recommendations. To show the effects of time decay on recommendations the authors use three patterns of time decay which are *concave time decay function*, *convex time decay function*, and *linear time decay function*. According to the result, the algorithm with time decay provides better recommendations if the value of the time decay coefficient is chosen properly.

4.2 Prepare Background and Context

Another important deficiency in calculating the rating score of a product is the time factor. Most of the existing rating score algorithms do not take into account the time of users' ratings. But as we mentioned above, time is an important factor in some e-commerce platforms, and it is believed that more recently reviewed products better appeal to customers' needs. Namely, hot ratings give us more reliable and valuable information about products. For these types of reasons, in this section, we first

introduce the concept of time decay by explaining its mathematical definition then we calculate a product rating score based on time decay.

In the method to be explained, we assume that a record is a quartet in which user *u* rate item *i* with *r* on the t^{th} day as shown in Table 4.1.

4.2.1 Time Decay of a Rating

The more current rate on a product by a user, the more current information for us about the quality of that product. To calculate the currency of a rate, we use the formula of the motion at constant or uniform acceleration. As it is known that acceleration is the rate of change of velocity of an object. It is so common in physics and daily life that some basic equations are derived to work out the situations in which acceleration is constant. As it is known the position equation for the constant acceleration is as follow:

$$d = \frac{1}{2}\alpha\, t^2 \tag{4.1}$$

Where d indicates the position, α is the acceleration and t is the time. In the following sections, we indicate the currency of a rate as a position. Thus, our equation will be as follow:

$$c = \frac{1}{2}\alpha\, t^2 \tag{4.2}$$

According to Formula (4.2), the currency, or we can say that the importance of a rate increases as the time increases. If the currency-time data for such a product were graphed, then the resulting graph would look like the graph as below:

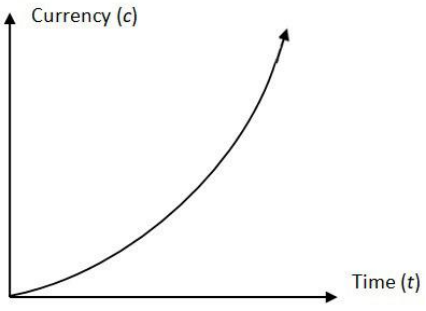


Figure 4.1 Currency-time graph

According to the graph in Figure 4.1, as the date of a rate gets closer to the current day, the importance of the rate increases. Let's show it with a real example on our dataset.

4.2.2 Calculation Rating Score of a Product Based on Time Decay

Let's assume that we get the product 2 from our real dataset.

Table 4.1 Information about product 2

Itemid	Userid	Rating	Day_Distance	Date
2	2244178	4	969	2002-02-27
2	66286	4	931	2002-01-20
2	1	5	3358	2008-09-12

Where *Day_Distance* indicates the difference of the date of the rate to the date of our database is created (which is 1999-07-04). The normal average rating score of item 2 is 4.333 but we want to calculate based on time decay. For this purpose, we first calculate the acceleration of our dataset according to the currency value which is set to over 100 in this study. The equation of the acceleration is as follow:

$$\alpha = 2c \div (t_1 - t_2)^2 \quad (4.3)$$

It is assumed that the currency of any rating is from 1 to 100 and time is the difference between the date of our database is created (which is the first day of our dataset obtained, t_2) and the date of the today (which is the last day of our dataset obtained, t_1). Thus, our α value will be as follow:

$$\alpha = 2 * 100 \div (2011\text{-}06\text{-}16 - 1999\text{-}07\text{-}04)^2 = 1.049$$

Now we can calculate the rating score of item 2 based on the weighted average with the currency of the rates. The equation will be as follow (4.4):

$$\check{r} = \frac{\sum_{u=1}^{n} cr}{\sum_{u=1}^{n} c} \quad (4.4)$$

Where ř indicates product rating score, c is the currency, and r is the rating of each user to the related product. Thus, we can calculate the weighted average of the ratings according to the currency of each rating. The currency of each user's rating is as follow:

Table 4.2 Currency of each rating

Userid	Rating	Day Distance ($t_1 - t_2$)	Currency ($(1/2) * \alpha * t^2$)
2244178	4	969 (2002-02-27 – 1999-07-04)	$= (1/2)*1.049*(969)^2=4{,}92$
66286	4	931 (2002-01-20 – 1999-07-04)	$= (1/2)* 1.049*(931)^2=4{,}54$
1	5	3358 (2008-09-12 – 1999-07-04)	$= (1/2)* 1.049*(3358)^2=59{,}14$

Thus, the weighted average of the ratings based on time decay is as follow:

$$\check{r} = \frac{(4{,}92 * 4) + (4{,}54 * 4) + (59{,}14 * 5)}{4{,}92 + 4{,}54 + 59{,}14} = 4.86$$

As you can see from the result, the rating score of the product has increased. The reason is that user 1 gives "5" star to the item and the date of the rate is more current than others. The other two users gave rates in 2002 but user 1's is in 2008. The difference in the date of the rates is about 6 years. This time difference is really important to us when we think about some e-commerce platforms such as hotels, restaurants, travel agencies, and other service-based companies. There can be many reasons for the increase in the rating score of the product 2. If we assume that the product 2 as a hotel, the owner of the hotel may have changed or the hotel could be renovated or the hotel's service policy may have changed, etc.

4.2.3 Calculation Rating Score of a Product Based on Helpfulness Votes

In order to compare our models (trust-based and time-based model) with traditional models, here we want to calculate the rating score of products based on helpfulness vote which is used in some e-commerce platforms.

Reviews are another quality assessment of products on e-commerce platforms especially in making the customer's final decision. Some algorithms calculate the rating score of items based on the number of reviews since the popularity of product increases by the number of reviews. But because the majority of reviews are fake, weak, or meaningless nowadays some e-commerce platforms use a voting system for the review rating such as amazon, TripAdvisor, booking, etc. In such systems, customers can vote the reviews if they want and, in this way, it is easier for the algorithms to determine the quality of the reviews. In our study, in order to understand the effect of the reviews on the rating score of items, we try to calculate the weighted rating score based on the helpfulness votes.

As seen in Table 4.3, the sixth column is the helpfulness vote of the review. We have five helpfulness categories, and these are 'Very Helpful', 'Helpful', 'Somewhat Helpful', 'Show', 'Not Yet Rated'.

4.2.4 Calculation Rating Score of a Product Based on True Bayesian Estimate

In order to compare our models (trust-based and time-based model) with traditional models, here we want to calculate the rating score of products based on True Bayesian Estimate (TBE) which is used in some e-commerce platforms.

To get healthy results when we calculate the average score of an item, it is important to regard the number of rated users to that item. We know that the more experienced item by users, the more known about the quality of that item for the other users. For instance, for a movie rating, it is not the same for the same average rating for two movies which one is rated by just two users and the other one rated by one million users. Of course, knowing the quality of the second movie will be healthier because of rated by a higher number of users. That's why web sites like IMDB use the TBE in order to assess the quality of movies.

True Bayesian Estimate is commonly used for forecasting weighting value in order to get logically accurate rating results. Actually, it is based on Bayesian Estimate but specifically utilizes the number of users' rates and sometimes other features. Let's look at the Formula (4.5):

$$Weighted\ Average\ (WR) = \{\left(\frac{v}{v+m}\right) * R + \left(\frac{m}{v+m}\right) * C\} \tag{4.5}$$

Where:

WR = weighted average rating for the item,

v = number of users who rated the item,

m = minimum number of users to be calculated by the formula,

R = average rating of the item,

C = Average rating of all items by the formula.

4.3 Experimental Results

In this part, we carry out several experiments in order to verify the quality of the proposed recommender model based on trust value and time decay. For this purpose, we perform 3 different methods, i.e., product rating score based on trust values, time decay, and both, and then we compare with a normal average score.

In this section, the dataset we use to evaluate the proposed algorithm is a real e-commerce dataset extracted from Epinions in June 2011. It is available at

http://liris.cnrs.fr/red/. It is worked on two datasets. The first one contains information about reviews from users on items and the second one contains trust relationships between users.

Table 4.3 Appearance of the rating dataset

Review id	User id	Rating	Item id	Date	Review rating
51902	182	4	43286	2001-06-21	Very Heplful
557406	236320	5	83979	2000-01-29	Helpful
557411	2377344	5	43032	2004-11-07	Somewhat Helpful
557415	237344	4	166791	2004-11-05	Show
510717	235	5	158368	2000-11-16	Not Yet Rated

The rating dataset contains review id, user id, item id, user's rating, between 1 and 5, and the date of the review. There are 1127673 reviews which 113629 users have at last one rating. Table 4.3 shows that the user 182 has a review with id 51902 and gives 4 points for the item 43286 on the date 2001-06-21. It is a .csv file and 62.0 Mb.

Table 4.4 Appearance of the trust network dataset

Trustor id	Trustee id	Value
22	434	1

The trust network dataset shows which user trust to whom, only positive values appear in the dataset and there are 538392 trust values which 47522 users have at last one trust relation. Table 4.4 shows the user 22 trusts the user 434. The value of 1 indicates that the user trusts to another one. It is a .csv file and 10.2 Mb.

Our dataset contains 131228 users, 317775 items and 1127673 reviews, namely our dataset has 0.003% sparsity. Our algorithms are executed on the Jupyter Notebook with Python version 2.7.11.

4.3.1 Rating Score Based on Trust Values

Table 4.5 Trust values of some users

User id	Trust Values
1	14.380358249
2	10.348435726
3	3.243809321

As we explained in Chapter 3, after applying the formula in (3.7) on our trust network dataset we get the results as it is shown in Table 4.5. Approximately, user 1 has 14, user 2 has 10, user 3 has 3 trust value, and so on, respectively. According to the result, we can say that user 1 is the most trustworthy user among the three users. As it is known, most of the existing recommender algorithms calculate the product rating score by calculating the average of all ratings of the users who rated the related item. But we calculate a weighted average based on the trust value of each user who rated the product. That is to say, user 1 affects the rating score of a product more than user 2 if they both rated to the related product [46].

$$\check{r} = \frac{\sum_{u=1}^{n} tr}{\sum_{u=1}^{n} t} \tag{4.6}$$

Where ř indicates product rating score, t is the trust value of the user, and r is the rating of each user to the related product. Thus, we can calculate the weighted average of the ratings according to the trust value of each user. Results are as follows:

Table 4.6 Average ratings based on trust values

item id	Number of users	Average of ratings	Average of ratings based on trust
1	2	4.0	4.0
2	3	4.333	4.965
3	9	4.555	4.117

According to Table 4.6, the rating score of item 1 does not change because of the given the same rate by the rated users. But for item 2, there is a great difference between average rating (AR) and weighted average rating based on trust values (WARTV). Most probably one of the users who have a more trust value gives a rate to the item more than average. The same for item 3 but this time one of the users who has a more trust value gives a rate to the item less than average. Let's examine one of these items in detail.

Table 4.7 Trust values of each user who rated item 2

User id	User's rating for item 2	Trust values of Users
1	5	14.380358249
66286	4	0.325909815
244178	4	0.201582285

As seen in Table 4.7, user 1 has great trust value more than other users. Because of this reason, even the other two users give 4 rates for item 2; item 2's rating score is almost 5. Another important output of the research is that the average difference between AR and WARTV decreases as the number of users increases. It means that items rated by too few people are affected easier by the fake accounts but according to our dataset, rating score of items rated by more than 100 people have almost the same rating score based on trust values. Let us see the average difference between AR and WARTV when the number of rated users increases on 1.000 items in our dataset.

Table 4.8 Average difference between AR and WARTV

Number of users	Number of items	Average Difference
2-10	339	0,32080
11-50	223	0,27434
>50	195	0,19845

As seen in Table 4.8, as the number of users who rated the products increases, the average difference between AR and WARTV decreases. Also, the number of items rated by more than 50 users decreases when we execute the algorithm on more than 1000 items. Of course, these results are just for 1000 items and when we increase the items, the results emerge clearer. Actually, when we continue to calculate the rating score of the products rated by more users, we see that the average difference between AR and WARTV comes closer, almost 0.

4.3.2 Rating Score Based on Time Decay

Table 4.9Average ratings based on time decay

item id	Number of users	Average of ratings	Weighted Average of ratings by TD
1	2	4.0	4.0
2	3	4.333	4.862
9	31	4.161	3.140
17	2	2.5	3.936

As seen in Table 4.9, after applying Formula (4.4) on our dataset, the rating score of each item changes according to the ratings' date of users. Some rating scores of the items increase while some decrease. That is to say, there is no regular structure.

4.3.3 Rating Score Based on Trust Values and Time Decay

In order to find rating score of products based on TV and TD, we apply Formula (4.7) on our dataset.

$$\check{r} = \left\{ \left(\frac{1}{2} \frac{\sum_{u=1}^{n} tr}{\sum_{u=1}^{n} t} \right) + \left(\frac{1}{2} \frac{\sum_{u=1}^{n} cr}{\sum_{u=1}^{n} c} \right) \right\} \tag{4.7}$$

Dividing one-half means that each method will affect the results equally.

Table 4.10 Average ratings based an time-decay and trust value

item id	Number of users	Average ratings	With TV	With TD	With TV and TD
1	2	4.0	4.0	4.0	4.0
2	3	4.333	4.965	4.862	4.913
9	31	4.161	4.388	3.140	3.764
17	2	2.5	3.707	3.936	3.821

As seen in Table 4.10, the weighted average rating score of products based on both methods is a balance between two other methods. This may be because of some users' having a high trust value, but their comments are too old or vice versa.

4.3.4 Rating Score Based on Helpfulness Votes

When we calculate a rating score of a product, we get all the ratings with 'review rating' for that product. Then, in order to find the weighted average rating, we multiply each rating of that product with the review rating. Namely, if the rating has a 'very helpful' review rating, we multiply the rating with 5 or if the rating has a 'helpful' review rating, we multiply the rating with 4 and so on. The output of some items' rating score can be seen in Table 4.11.

Table 4.11 Average rating based on helpfulness votes

Item id	Number of users	Average rating	Average rating based on helpfulness votes
3	9	4.555	4.523
9	31	4.161	4.21
12	3	3.666	3.800
15	34	4.323	4.307

As seen in Table 4.11, there is no regular decrease or increase in value between ARS and weighted average rating based on helpfulness votes (WARHV). In other words, while the rating score of some items increase, other decreases.

4.3.5 Rating Score Based on True Bayesian Estimate

We execute the algorithm according to the Formula (4.5) to see the difference average and weighted average based on True Bayesian Estimate. We got the 'm' value as 1 and the average ('C') of all products of this category was equal to '3.68863983712'.

Table 4.12 Average rating based on True Bayesian Estimate

Item id	Number of users	Average rating	Average rating based on TBE
32820	1	5.0	4.344
218694	2	3.0	3.229
61559	66	4.424	4.413
34328	99	4.040	4.036

As seen on the Table 4.12, average rating results come close the general average ('C') because of the formula. That is to say, if the average score of an item is less than average score of all items (which is overall average and equal to 3.688), the result is drawn towards the overall average because of the formula. Similarly, if it is higher than the general average, the result is drawn down.

4.3.6 Comparison of all Weighted Averages with each other

Table 4.13 Comparison the difference between average and all other weighted averages

Number of users (Range)	Number of items	Trust Values	Time Decay	Helpfulness Votes	True Bayesian Estimate
1-5	249	0.302	0.163	0.040	0.004
5-10	90	0.373	0.228	0.035	0.016
10-20	100	0.269	0.244	0.043	0.000
20-50	123	0.278	0.230	0.035	0.006
50-100	83	0.248	0.227	0.022	0.000
100-200	54	0.174	0.240	0.020	0.003
>200	36	0.125	0.241	0.012	0.002

According to the results in Table 4.13, it seems the True Bayesian Estimate has the best results. But if we take into account the formula of TBE, the mentality of the formula is already pulling the rates of the items to the average. Because of this reason, the difference between the average and TBE is almost close to zero. But when we think of the increasing range, it seems not regular. Namely, when the range goes up the difference is not close to zero. It changes range to range. If you realized, Helpfulness Votes is the better than True Bayesian Estimate if we take into account the increasing range. Actually, this has not surprised us because of getting the help of people but it is too hard to get helping of people for voting the reviews. If you look at the Time decay results, we see there is no regular motion. Time Decay weighted average totally depends on the time of each user's rating date. Indeed, Time Decay weighted average would be better for some types of e-commerce platforms especially hotels, websites and other service-based platforms because of their nature. The weighted average based on trust values is also almost regular. Its results come to close zero according to the increasing range. It differs from other weighted averages as Time Decay.

4.4 Conclusion

In this chapter, we tried to improve the quality of the product rating score based on the trust values of users and time decay of the date of users' ratings. First, we introduce the concept of PageRank by giving its mathematical definition and redefine for revealing the relationship between users in Chapter 3. We tried to reduce the effects of fake accounts on the rating score of products by using the trust value of each user. As seen in Table 4.8, the normal average rating score of products comes close to the trust values based average as the number of users who rated the related products increases. This result shows us that our algorithm is on the right track. Then, we introduce the concept of time decay by giving its mathematical definition and redefine for reducing the effects of old ratings when determining the rating score of products. After that, we apply both methods to the dataset. In this way, we break down the power of one method on the results since a product can be rated by trustful users, but their ratings' date may be too old or vice versa. But if a product is rated by trustful users and their ratings' date is up to date enough, we can say that the product gets the value it deserves. Lastly, we compared average ratings with all weighted average. According to the results, Due to

the structure of the formula, the True Bayesian Estimate seems the best method. However, when we consider the products one by one, some products get a high rating score while others get low, but this is due to the structure of the formula, not the quality of the product. Apart from that, finding rating score of an item based on helpfulness votes gives good results, but this method requires extra user assistance and there is not enough data for many e-commerce platforms.

5. THE IMPACT OF TEXT PREPROCESSING ON THE PREDICTION OF REVIEW RATINGS

With the increase of e-commerce platforms and online applications, businessmen are looking to have a rating and review system through which they can easily reveal the feelings of customers related to their products and services. It is undeniable from the statistics that online ratings and reviews charm new customers as well as increase sales by means of providing confidence, ratification, opinions, comparisons, merchant credibility, etc. Although considerable research has been devoted to the sentiment analysis for review classification, rather less attention has been paid to the text preprocessing which is a crucial step in opinion mining especially if convenient preprocessing strategies are found out to increase the classification accuracy. In this chapter, we concentrate on the impact of simple text preprocessing decisions in order to predict fine-grained review rating stars whereas the majority of previous works focused on the binary distinction of positive vs. negative. Therefore, the aim of this section is to analyze preprocessing techniques and their influence, at the same time explain the interesting observations and results on the performance of a five class-based review rating classifier.

Especially over the past decade, fast-growing e-commerce platforms have begun to dominate the entire business world. Thanks to the many options provided by these platforms, customers started to feel more comfortable than traditional commerce by finding experienced products, which are reviewed and rated by too many people who are expressing and sharing their own feelings and thoughts about any products. Thus, customers' opinions began to play a major role in purchasing decisions, business intelligence, and keeping any product or service available. Many studies and surveys conducted by companies and researches have proved this situation that sentiment analysis is a constantly growing area in recent years [70]**.** Holleschovsky and Constantinides [6] show that 98% of the sample research population read reviews before making a purchase and 60% of them read often or quite often. Last ReviewTrakers online survey shows that 6 out of 10 consumers look to Google for checking online reviews before visiting a business [71]. Tripadvisor indicates that travelers rely on reviews and opinions from other travelers before booking their trip

[72]. Therefore, the field of sentiment analysis, which is also called opinion mining suddenly, became a popular research field because of providing opportunities to the companies wanting to know the pros and cons of their products or services to identify new strategies as well as make crucial decisions.

Specifically, sentiment analysis in reviews is the process of analyzing, monitoring, and categorizing thoughts, opinions, or feelings from an unstructured text about a product or a service, especially in e-commerce platforms. Namely, it works on unstructured review text to find useful information for business intelligence. There are a couple of steps for the text classification such as preprocessing, feature extraction, feature selection, and classification.

Although sentiment analysis is a relatively new area of computer science, there are considerable researches except for the importance of text preprocessing on classification performance. Therefore, in this section, we specifically focus on the role of various text-preprocessing stages which are the initial processes in sentiment analysis to demonstrate the effects by experimental results on the performance of a five class-based review rating classifier. Generally, preprocessing consists of some methods such as tokenization, lemmatization, stemming, lowercase conversion, replacing negation, reverting repeated letters, expanding acronym, removing stopwords, numbers, URLs, punctuations and special characters, etc.

There are few types of research on predicting fine-grained rating stars in review texts which is a challenging task because of the low probability of estimation and use of similar words for closed classes by users. Thus, it is important to know which preprocessing method will increase the classification accuracy and how and why it affects the results.

The rest of this chapter is organized as follows. After the introduction, Section 5.1 presents some of the recent works especially focused on preprocessing techniques for text classification. In Section 5.2, we explain some details about each preprocessing method and give some specific examples about the related area. Section 5.3 introduces a real-life dataset used in our experiment and Section 5.4 reports some experimental outcomes and evaluates the results. Finally, we conclude and discuss in Section 5.5.

5.1 Related Work

After the sentiment analysis is really drawn a great deal of attention among data mining researches in the last decades due to the charming commercial returns, researches related to this field started to increase, particularly on classification models aiming to improve the sentiment classification accuracy. In this section, we specifically focus on some recent related researches which deal with different types of preprocessing methods to improve the performance of a classifier.

When we look at the recent studies in general, some of them indicate that certain preprocessing methods have a great effect on the performance of classifiers while some of them state that they are only slightly better or do not show any effects or even worse. Below is a close look at some of these studies.

Sharma et al. [73] investigate the impact of preprocessing on four different Twitter text data i.e. sports, politics, entertainment, and finance. According to the results, removing stopwords, URL links, punctuations, and converting lowercase increase the classification accuracy of the Twitter sample data.

Ghag et al. [74] investigate the impact of removing stopwords on several sentiment classification models using the movie document dataset. According to the results, while removing stopwords has a great effect on the classification accuracy for the traditional sentiment classifier, there is no significant change for the other classifiers such as the Average Relative Term Frequency Sentiment Classifier, Sentiment Term Frequency, Inverse Document Frequency, and Relative Term Frequency Sentiment Classifier.

Jianqiang and Xiaolin [75] investigate the impacts of preprocessing techniques for the performance of sentiment classification on five Twitter datasets. According to the experimental results, while removing URLs, numbers and stopwords have a little effect, expanding acronyms and replacing negation have a huge impact on the classification accuracy and F1 measure for the classifying Twitter texts.

Srividhya and Anitha [76] investigate some preprocessing techniques whether they have an impact on the classification accuracy on the Reuters dataset. According to the results, removing stopwords, stemming and TF/IDF have a great effect on the performance of classification.

Camacho-Collados and Pilehvar [77] study on the role of simple pre-processing techniques on the performance of Neural Text Classifier using tokenizing, lemmatizing,

lowercasing and multi-word grouping. According to the research, using simple tokenization affects more than complex preprocessing techniques such as lemmatization or multi-grouping. The research also shows that the effects of pre-processing changes according to the size of the training data used.

Ghag et al. [78] work on some pre-processing techniques for optimizing sentiment classification. For this purpose, they focus on some rules to handle apostrophe and punctuation symbols, unlike traditional pre-processing techniques. According to the results of the research, the accuracy of classification increases by the proposed pre-processed data, and the elimination of the stopwords decreases, unlike traditional sentiment classification.

Gull et al. [79] use pre-processing techniques in order to analyze useful political structured content. For this purpose, they get the data from Twitter, and then they clean tweets especially useless emoticons, punctuations and URL links using some pre-processing techniques. After that, they extract hash tags and change of direction indicators on the selected parsed tweets for classification. According to the results, SVM is better than Naïve Bayes for tweet classification.

Jianqiang [80] works on the preprocessing techniques in order to show their effects on Twitter Sentiment Analysis especially cleaning tweets from URL links, stopwords, repeated letters, negation, acronym, and numbers. According to the authors, some pre-processing techniques hardly change the accuracy of sentiment classification such as removing URL links, numbers, and stopwords.

Safeek and Kalideen [81] work on spell correction and emoticon analysis in order to get suitable data for Sentiment Analysis on Facebook data. According to the authors, writing “happppyyyyy” is more strength than “happy”. Namely strength of the word is defined how many times a character occurs in a word.

Vijayarani et al. [82] explain various pre-processing techniques in their research especially stopwords elimination, stopwords removal methods and stemming algorithms for classification processes such as truncating methods, statistical methods, and mix methods.

Hemalatha et al. [83] apply some preprocessing techniques to be ready for giving a text as an input to any Machine Learning algorithms. For this purpose, they remove URLs,

special characters, question words and repeated characters in order to help a given document to be ready as an input into any Machine Learning algorithms.

To improve mining process, Katariya and Chaudhari [84] suggest using the text data with the use of side information such as web logs, links in the document and meta-data. Therefore, they get text documents from different sources and then they apply preprocessing techniques on the obtained information. The research shows that the domain specific application is more proper for text mining.

Singh and Kumari [85] study on the effects of preprocessing and normalization on the short text like tweets. They especially evaluate the effects of slang words in a tweet to show how they change the accuracy for a better sentiment classification.

Nayak et al. [86] work on two basic stemming algorithms to reveal the pros and cons of each of them. According to the authors, MF Porter's algorithm leads to a large degree, therefore it finds incorrect stem whereas the Krovetz algorithm is ineffective with a large document.

Krouska at al. [87] execute some preprocessing techniques on three different Twitter datasets. According to the results, using appropriate feature selection and representation of the dataset may increase the classification accuracy in Sentiment Analysis such as 1-to-3 grams perform better than other representations and feature extraction.

Zin at al. [88] show the effects of various preprocessing strategies such as stopwords, numbers, punctuations, etc. with experimental results on online movie reviews. Their study proved that preprocessing affects the performance of the classification in a good way especially on the SVM with non-linear kernel.

Pomikálek and Řehůřek [89] study on preprocessing parameters such as stopwords list selection, stemmer selection, and tokenizers in order to compare them on three text data sets and they show how these parameters affect results. According to their results, the term weightier "ntc" (tf.idf) works best with the shorter documents whereas term frequency "atc" performs better with longer documents.

Schofield et al. [90]investigate the effects of preprocessing in sentiment classification. According to the results, the influence of many common preprocessing techniques such as stemming, removing stopwords have little effect or even negative effects. They suggest that instead of applying the common preprocessing techniques on the text data, it can be more efficient to decide to preprocess techniques according to the application.

Fan and Khademi [91] concentrate on the effects of top frequent words in raw text reviews and top frequent words/adjectives after part of speech analysis results. According to the results, raw data has almost equal power for different feature generation methods whereas determining words and adjectives after part of speech can remove informative features out.

There are not only researches about text preprocessing in English but also other languages. One of them is the research of Duwairi and El-Orfali [92] who investigated the effects of text preprocessing methods on the classifiers' accuracy in the Arabic language. According to the results, stemming and removing stopwords affect the performance of the classification badly for the movie review texts while slightly improve for the political texts. The other one is Saad's research [93], which investigates the effects of text preprocessing on Arabic text classification applying term weighting schemes, morphological analysis, namely stemming and light stemming. According to the experimental results, light stemming with term pruning works very well for feature reduction and weighting schemes affect the accuracy of the distance-based classifier. As in the Arabic language, there are some challenges in some languages because of having very complex morphology as we compare to the English language. For this reason, preprocessing is very important for text mining. One another study in a language other than English is Uysal and Gunal's [94] research which shows the effects of preprocessing techniques on two different text domains and languages, namely Turkish and English. For this purpose, they use all potential combinations of preprocessing strategies by thinking of several ways. According to the results, using proper combinations of preprocessing strategies provide successful accuracy on text classification depending on domains and languages studied.

5.2 Prepare Background and Context

Online review websites play a vital role in all aspects of the business world especially with the increase in e-commerce platforms. Nowadays, most of the review-based e-commerce websites like amazon.com, TripAdvisor, booking.com, alibaba.com, etc., are extensively dominate the market. The best parts of these kinds of platforms are that customers can comment on products, rate products, and can easily reach to the other reviews about products written by other users. For this reason, it is important to analyze

reviews and ratings in order to determine new strategies and provide better service to customers.

Preprocessing is the first step of the sentiment analysis after getting a dataset. We apply this process to clean and prepare texts for sentiment classification because texts particularly written by users are unstructured. Namely, unstructured texts usually have lots of noisy, unnecessary, useless information such as repeated words, numbers, punctuations, Html tags, URLs, scripts, advertisements, stopwords, abbreviations, emoticons, slang words, misspelling, shortcuts, specific terminology, etc. Because of treating each word as a dimension in the feature set, having all unnecessary words cause the models to be confused and loss of time. On the other hand, cleaning the text from noisy data may increase the performance of classifiers as well as accelerates the classification process.

Even we can't show all the details in this research due to the space limitation, preprocessing contains very different steps such as tokenization, removing emoticons, punctuations, URLs, stopwords elimination, stemming, lemmatization, expanding abbreviation, lowercasing, multiword grouping, word correction, the strength of words, weighting scheme and removing common words. Although there have been remarkable researches on this field, finding the best preprocessing method is still an open issue. Researchers show that the best preprocessing methods change according to the application. Therefore, in this study, we concentrate on review texts specifically related to restaurants. Below are some of these preprocessing methods step by step.

5.2.1 Tokenization

Tokenization can be defined as splitting up a text into the desired list of practical pieces called tokens such as words, phrases, symbols, or other units, or even whole sentences in order to work on the text more effectively. It is considered an important process of Natural Language Processing because of being an input for the next processes. We use whitespace, punctuations, and sometimes line breaks to get tokens. In most cases, we use whitespace.

There are a couple of tokenizers in Natural Language Toolkit (nltk) which is a platform to work on human language data. One of them is the Regexp tokenizer. This tokenizer split a sentence using regular expression for matching tokens. For instance, if we use

RegexpTokenizer(“[/w`]+”) for a sentence like, “We’ll go on a picnic tomorrow.”. We will get a result like: [“We”, “ll”, “go”, “on”, “a”, “picnic”, “tomorrow”].

The second one is TreebankWord tokenizer. This tokenizer split the sentence according to the regular expression but treats the punctuations as a word, so it splits commas, apostrophes, quotation marks, etc. For instance, if we use TreebankWordTokenizer() for the same sentence above, we will get a result like: [“We”, “ ’ ”, “ll”, “go”, “on”, “a”, “picnic”, “tomorrow”, “.”].

The third one is WordPunct tokenizer. This tokenizer split the sentence according to this \w+|[^\w\s]+ regular expression. For instance, if we use WordPunctTokenizer() for the same sentence above, we will get a result like, [“We”, “ ’ ”, “ll”, “go”, “on”, “a”, “picnic”, “tomorrow”, “.”].

As it is seen we got the same result as TreebankWord tokenizer, of course for the given sentence. There are more tokenizers in nltk tool to use according to the need.

5.2.2 Effect of Emoticons, Removing Punctuation and Urls

Most of the time, it does not make sense to treat emoticons and punctuations as a token for the sentiment classification. Thus, removing emoticons (e.g. :-), :), :-), :-(are frequently used in social media and messaging applications), and punctuations (^!”#$%`()*+-/:;<=>?@[\\]|~{}) increase the accuracy of the classification because of being treated as a dimension in the feature set for each word. But sometimes especially emoticons can have a slightly good effect on the sentiment score according to the searching area [95]. The research in [96] shows that the importance of emoticons on polarity sentiment classification especially in social networks is undeniable and their popularity is getting higher and higher.

In most literature, URLs do not have any information to analyze regarding sentiments in texts. For instance, when considering the following sentence, “I hate all those disgusting meals from www.mydeliciousmeals.com if you want better, you can click www.besteverdinner.com” actually the review is negative but because of the words in links, it may become a positive review. Thus, researchers want to remove URLs from texts to avoid such situations. But for some specific application URLs can be effective for providing insights about the text in a way that is not easily obtainable from the context.

5.2.3 Expanding Abbreviation and Acronyms

We can say that abbreviation is a shortened form of words and most of the time, their full meaning is given at the first used place. We widely use the abbreviation to avoid repetition of words that are used too many times in a text and to save space. Usually, they are formed by getting the first few letters such as Aug. for August, CA. for California, univ. for university, etc. Sometimes they are formed by omitting letters such as TX. for Texas, St. for Street, Rd. for road, Dr. for Doctor, etc.

The difference between abbreviation and acronym is that acronyms are formed by getting the first letters of each word of the phrase such as A.S.A.P for as soon as possible, PA. Personal Assistant, Lol. for Laugh out loud, TY. for Thank you, NP. for No problem, FBI. for the Federal Bureau of Investigation, AI. for Artificial Intelligence, etc.

Expanding abbreviations and acronyms are important to understand the contexts in text mining. Compared to the past, the problem of abbreviation and acronym has attracted relatively more attention in text mining especially after increasing the number of messaging applications such as WhatsApp, Viber, Tango, Line, etc. and social media platforms such as Facebook, Twitter, Instagram, Snapchat, etc. For instance, the acronyms such as Omg (Oh my God), Lol (Laughing out Loud), 2moro (Tomorrow), B3 (Blah, Blah, Blah), ASL (Age / Sex / Location), F2F (Face to Face), BTW (By the Way), XOXOXOX (Hugs, kisses,…), PAL (Parents are listening), BRB (Be right back) are just some of them and they are used too much in daily life conversations. The ability to expand abbreviations and acronyms is crucial for many natural language processing applications and to find out the information contained in documents for information retrieval [97].

5.2.4 Word Correction and Multiword Expressions

Word correction which is also called misspelling checking is a method that identifies misspelled words in order to change with their most possible similar words. For this purpose, the misspelled word is checked whether it is presented in the dictionary or not. If it is not, the algorithm tries to provide the best similar word of it [98]. There are some types of misspelling such as keyboard errors (“yur” – “your”, “allways” – “always”,

etc.), cognitive errors (“piece”–“peice”, “sipritual”-“spiritual”, “freindly”-“friendly” etc.), phonetic errors (“calander”-“calender”, “katalog”-“catalog”, etc.), etc.

Bertoldi et al. [99] empirically work on the effects of misspelled words to show the performance of the Machine Translation. According to the research results, performance is related to the noise rate and the noise source affects the capability of Machine Translation.

There are some tools (e.g. nltk, word2vec, python grammar-check) for checking the misspelled words in texts according to the different languages. Some of them provide the best option, while others offer more than one alternative to the users classifying by types of misspelling. And some of them also check grammatical mistakes by examining everything that forms incorrect use of a person to subject-verb agreement.

Another important challenge in Natural Language Processing is multiword expressions which are generally difficult to trace from their individual words. They can be metaphorical expressions such as “killing time”, “broke someone heart”, “time is a thief”, etc. or verbal idioms such as “give away”, “made out”, “take off”, “come along with”, etc. or phrasal verbs or stereotyped comparisons such “as nice as pie”, “swear like a trooper”, “cold as stone”, etc. [100], or some well-known group of words such as “United Kingdom”, “Galaxy note 9”, “Citizen of Humanity”, etc. and so on. Thus, tokenizing such multiword expressions for text mining causes words to lose their meaning in the sentences. Consequently, getting these types of multiword as a single word can increase the performance of the classifier. There are some studies specifically on this topic [77], [100], [101] to show their effects on text mining. The study [101] investigates two empirical methods to integrate multiword expressions in a real constituency-parsing context.

5.2.5 Stopwords Elimination

In general, stopwords mean the most common words in a language, for us in English such as “and”, “an”, “at”, etc. which are considered unnecessary and useless in text mining applications. These words can be pronouns (I, me, my, mine, myself, etc.), prepositions (on, in, next to, behind, under, around, etc.), conjunctions (once, until, when, why, since, after, etc.), articles (a, an, the), auxiliary verbs (be, do, have, will, can, may, etc.), etc. Most of the studies show that stopwords should be removed from

the corpus without losing valuable information before the feature selection because of their negative effects on the performance of the sentiment classifier. But sometimes removing the stopwords might reduce the accuracy of classification such as documents or texts related to prepositions, conjunctions, auxiliary verbs, etc. So, removing stopwords can make the matching impossible but as we said, generally due to reducing the size of the feature set comparatively, it has a good effect in text mining. Normally, researchers use compiled lists (Rainbow list, Van stoplist, Smart stopwords list, etc.) provided by text mining tools but sometimes researchers create and then use those predefined lists according to their application. In this study, we use nltk tools but also, we modified the stopwords list according to our text structure.

There are some methods to eliminate stopwords from a text. The basic one is using pre-compiled lists as we mentioned. The other one is finding the most frequent words which are not needed for matching in the texts. For instance, if you study on restaurants reviews, the word “food” or “meal” generally will not give meaningful results because these words are used in both negative and positive reviews. Maybe these words can be used with the combination of other words, namely bigrams or Ngrams combinations such as “terrible tasting”, “food tastes bad”, “never had a bad meal”, etc. Actually, this is another preprocessing method but for some researchers, it can be under the branch of stopwords elimination. One another method is selecting words that occurred rarely and not related to your texts. There are some additional methods that are examined and studied.

5.2.6 Stemming

The aim of the stemming is to take words in a way in which they occur in a text so that reduce them to root forms by removing of their affixes such as prefixes (cutting off the beginning of the word) and suffixes (cutting off the end of the word) according to some grammatical rules. In this way, they can be used as an indexing unit in the related research area. Although stemming algorithms in most application tools are commonly developed for English, there is a need for appropriate editing according to the language being studied because of differences in language structure. Nowadays, many different algorithms can be also used for some other languages.

Stemming is applied for a couple of reasons. One of them is to reduce the derivatives of the same root words to the common representations to increase the performance of classification. One of the other reasons is to reduce the size of the feature set so that the number of dimensions is reduced.

We can apply the stemming to derivatives of the word such as number (cat, cats), tense (play, played, playing), gender (actor, actress), pronouns (I, me, my, mine), person (hate, hates), aspect (become, became), etc. For instance, the words select, selected, selecting, selects all can be stemmed to the word “select”. As it is seen, we cut off the end of the words which are semantically related to their root form. In this way, we reduce the number of words in memory space and save time.

When applying stemmer, we should consider some points which are important and required for a powerful natural language processing application. One of them is overstemming which occurs when the words have the same root but not having the same meaning. For example, “general” and “generation” can have the same root “gener”. Similarly, “organization” and “organs” have the same root “organ” and this situation decreases the accuracy of the classifier. One another is understemming which occurs in some stemmer algorithms. For example, the stemmer takes the words “cooks” and “cooked” and reduces to “cook”, while “cookery” can be reduced to “cookeri” or “absorbtion” and “absorbing” are stemmed to “absorpt” and “absorb”. This causes corresponding documents to not be returned.

Natural Language Toolkit platform uses a couple of stemmers such as PorterStemmer, LancasterStemmer, RegexpStemmer, Snowball Stemmer, etc. For example, while the PorterStemmer reduces the word “cookery” to “cookeri”, LancasterStemmer reduces to “cookery”. But if you want to use RegexpStemmer, you should determine your affix. For instance, when you use RegexpStemmer(‘ing’), it brings the word “cooking” as “cook” but you need to be careful in case the word has a prefix such as the word “ingrain”, it will be returned as “rain”.

5.2.7 Lemmatization

Both stemming and lemmatization are language preprocessing methods to provide that different versions of a word are not left out. Even they are closely related to each other, lemmatization is more complex than stemming because it reduces derivationally related

word form to its dictionary form categorized by a part of speech as well as by inflected form. Namely, stemming is applied without checking the position of the word in the sentence. So, if the user queries the plural and singular form of a word such as “mice” and “mouse” when the stemmer brings them as “mice”/“mic” and “mous”, the lemmatization brings “mice” and “mouse” both as “mouse”. By the time, for lemmatization we need to indicate the position of the word otherwise lemmatization gets the position of the word as “noun” by default. For example, when you use wordnet lemmatizer like, wordnet_lemmatizer.lemmatize(‘are’). Lemmatizer will bring it as “are”. For that, we should write as, Wordnet_lemmatizer.lemmatize(‘are’, pos=’v’). Then, lemmatizer will bring it as “be”. Let’s look at some words results after executing lemmatization.

Table 5.1 Difference between stemmer and lemmatizer

Words	**Porter Stemmer**	**WordNet Lemmatization (pos=verb)**	**WordNet Lemmatization (pos=noun)**
Constructing	construct	construct	constructing
Extracts	extract	extract	extract
Decided	decid	decide	decided
Took	Took	take	took
Information	inform	information	information
Clearly	clearli	clearly	clearly
Is	Is	be	Is

As it is seen in Table 5.1, according to our indication lemmatizer finds a base form of the words. Namely, lemmatizing means that converting the word to its dictionary form or morphologically related form. For example, for the sentence like, "I loved cats, dogs, frogs, and geese secretly". Lemmatizer will bring it as, “i love cat dog frog and goose secretly”. We informed the lemma function that “love” is a verb, “cats”, “dogs”, “frogs”, “geese” are noun and “secretly” is an adverb, and then we get the above result. Whereas if we use stemming for the sentence, it will return as, “I lov cat dog frog gees secret”. Namely, stemming returns the root of the word whereas lemmatizing returns dictionary form according to the position of the word in the sentence.

5.2.8 Lowercasing

Lowercasing is one of the first stages of preprocessing for text mining. All letters are converted to the lowercase to prevent case sensitivity. In this way, we can increase the performance of classifiers without considering the non-consistence of texts. Even though this simple preprocessing technique provides the easiest and important help to the classification, sometimes doing this might create some problems by increasing ambiguity. For example, Turkey is a country, but turkey is an animal or Opera is a browser, but opera is a musical play, so getting words in the lower case would be considered as identical entities for these types of words in text classification.

5.2.9 Removing Common Words

Removal of the common words does not guarantee that the accuracy of the classifier will be higher, but for most applications, it gives very good results. Common words and stopwords should not be confused with each other. Stopwords can be most common words but when we say common words here it means that they are found in almost each different class documents related to the studied field. So stopwords are almost the same for all studied field while common words are totally different for each studied field. For instance, the words “meal”, “dinner”, and “menu” can be the most common words for a restaurant corpus while the words “room”, “reception”, “bed” can be the most common words for a hotel corpus. As you realize, these words are not enough to find the differences between hotel or restaurant rating classes.

5.2.10 N-grams

Recently, many types of research on text mining and natural language processing have focused on Ngram. According to Ngram, it is not a coincidence that the words in a text are found more than once together. In other words, these words together give us a clue about the text summarization. In particular, the surprising effect of the text classification has been proven by many types of research. Of course, N-grams’s effect changes from research to research. For instance, according to the results of [102], Ngram works better on the shorter texts since the presence of words in shorter texts are more important than

longer texts. Namely, the value of a word loses its significance or value in a long text. Anyway, so what exactly do we mean by Ngram? Let's explain the Ngram with an example. If we want to use bigram for a sentence like, "An n-gram is a contiguous sequence of words in a text". After cleaning from the stopwords, the output of the program will be as, [('n-gram', 'contiguous'), ('contiguous', 'sequence'), ('sequence', 'words'), ('words', 'text')].

As it is seen, Ngram takes each word with the adjacent word. According to the frequency of these adjacent words, the content of the text is estimated, and the classification is made according to the result. If we had done the same example for trigram, we would get an output like, [('n-gram', 'contiguous', 'sequence'), ('contiguous', 'sequence', 'words'), ('sequence', 'words', 'text')]. As it is seen, at this time Ngram takes three consecutive words. And again, according to the frequency of these adjacent words, the class of the text is decided. According to the studied field, the number of the N can be changed but, in this study, we use the combinations of the Ngrams from one to three.

5.3 Dataset Description

In this paper, the dataset we use to evaluate the preprocessing methods is a real e-commerce dataset extracted from Yelp in June 2018. It is available at https://www.yelp.com/dataset. We work on two datasets. The first one contains full review text data including the User_id that wrote the review and the Business_id the review is written for and the second one contains business data including location data, attributes, and categories.

Table 5.2 Appearance of review dataset

Business_id	Date	Review id	Star	Text	User_id
9yKzy…	2011-01-26	fWKv…	5	My wife took me here on my birthday for breakf...	rLtl8Z…

The review dataset contains Business_id, the date of the review, Review_id, star, review text, and User_id. There are 229907 reviews, from which 43873 users have at last one review. Star column contains values ranging from 1 to 5. The value of "1" indicates that the user does not like the business at all, while the value of "5" indicates that he likes it

very much. If the ratings column contains a value of “0”, it indicates that the relevant user did not score the relevant product. The star column consists of 33.141% 5 star, 34.744% 4 star, 15.381% 3 star, 9.115% 2 star, 7.619% 1 star, respectively. Table 5.2 shows that the user rLtl8Z… has a review with id fWKv… and gives 5 stars for the business 9yKzy… on the date 2011-01-26. It is a .json file and 206.0 Mb.

Table 5.3 Appearance of business dataset

Business_id	Categories	Address	Name	Review_count	Star
qarob…	[Sandwiches, Restaurants]	891 E Baseline Rd\nSuite 102\nGilbert, AZ 85233	Jersey Mike's Subs	10	3.5

The business dataset shows business location data, category, its name, how many reviews it gets from users, and its average star rating. There are 11527 businesses which have at last one review. There are 509 different categories from 61 different cities. 11.034% of businesses 5 star, 15.151% 4.5 star, 23.178% 4 star, 22.874% 3.5 star, 13.097% 3 star, 8.763% 2.5 star, 3.493% 2 star, 1.474% 1.5 star and 0.936% received 1 star. 14.577% of the businesses are restaurants, 5.442% shopping, 5.231% food, 2.473% Beauty & Spas, 2.072% Nigthlife and the rest are other categories, each below 2%. Table 5.3 shows the business qarob… which reviewed by 10 people and got 3.5 stars on average and it is in the restaurant category. It is a .json file and 4.08 Mb. Our algorithms are executed on the Jupyter Notebook with Python version 3.6.5.

5.4 Experimental Results

In this section, we carry out several experiments in order to verify the effects of preprocessing methods on the performance of the classifier. For this purpose, we perform ten different methods, i.e., tokenization, effects of emoticons, removing punctuation and URLs, expanding abbreviations and acronyms, word correction and multiword expressions, stopwords elimination, stemming, lemmatization, lowercasing, removing common words and lastly Ngrams effects.

In order to get meaningful results, we created our feature set on 10000 restaurant reviews. Our aim is to analyze the effects of preprocessing methods when finding the star ratings of restaurants by analyzing the reviews. Star ratings range from 1 to 5. As

we mentioned above, the biggest challenge is to find the star rating of close categories because of using very similar words.

We use K Nearest Neighbors (KNN), Decision Tree (DT), Random Forest (RF), Logistic Regression (LR), Stochastic Gradients Descent (SGD), Naïve Bayes Classifier (NB), Support Vector Machine (SVM) classifiers of nltk to get accuracy results for all types of preprocessing methods. Over the created feature set, 10000 reviews are applied for training the classifier (Classifiers are applied for each rating category in equal number, namely, 2000 training reviews are selected from each rating category.) and 1000 reviews are applied for testing (200 testing reviews are selected from each rating category) in order to see the performance of the classifier.

To get stable results from the effects of a random selection of reviews, we run each experiment by selecting 20 times shuffled reviews for each preprocessing method. Namely, for each run, a different subset of reviews is selected from the pool of all available five rating categories.

5.4.1 Performance of the Classifiers Based on Different Tokenizers

In this part, we report the results obtained after tokenizing the text of the reviews as we mentioned above. In addition to those techniques, we also used WhitespaceTokenizer which is simply used for tokenizing the text according to the white space between the words. WhitespaceTokenizer method can be considered as the result without any preprocessing. In other words, at least this basic tokenizer form must be applied to the data before the other preprocessing method can be performed. Therefore, for the rest of the preprocessing methods, we choose space tokenizer to create a dictionary with the 1-to-3 n-grams in order to get directly simple effects of the methods.

Table 5.4 Performance of the classifier based on different tokenizers

Tokenizer	KNN	DT	RF	LR	SGD	NB	SVM
WhitespaceTokenizer() – Base Form	0.260	0.287	0.374	0.476	0.446	0.491	0.444
RegexpTokenizer("[w']+")	0.284	0.318	0.404	0.513	0.486	0.512	0.478
TreebankWordTokenizer()	0.281	0.303	0.366	0.510	0.485	0.518	0.465
WordPuncTokenizer()	0.266	0.338	0.403	0.503	0.478	0.518	0.476

As shown in Table 5.4 after running the methods 20 times shuffled reviews, there is no significant difference between RegexpTokenizer("[\w']+"), TreebankWordTokenizer(), and WordPunctTokenizer() tokenizers, all the tokenizers gave almost the same results. But we should also indicate that we observed the WhitespaceTokenizer() gives a little bit worse results on general average all the times rather than others when we run the program multiple times.

5.4.2 Classifier Performance Based on Replaced Emoticons and Removing Punctuations

In this part, we report the results obtained after removing emoticons and punctuations separately. We don't apply the removing URLs method due to a lack of the remarkable number of reviews as we observed from reviews text.

To see the effects of emoticons on the rating stars, we investigate the usage of emoticons in the text reviews. For this purpose, we create a simple word replacer in order to change emoticons to words. In this way, our program can find a relationship with a combination of words. For example, we replace the emoji ":)" as a "smile", ":(" as a "sad", ":-o" as a "surprised", etc. which are commonly used in messaging applications and social media.

Table 5.5 Classifier performance based on replaced emoticons and removing punctuations

Preprocessing Methods	KNN	DT	RF	LR	SGD	NB	SVM
Base Form	0.260	0.287	0.374	0.476	0.446	0.491	0.444
Replaced Emoticons	0.281	0.311	0.314	0.474	0.428	0.492	0.438
Removing Punctuations	0.276	0.289	0.379	0.474	0.435	0.491	0.436

As shown in Table 5.5 after running the methods 20 times shuffled reviews, the average accuracy result of replaced emoticons is sometimes worse than without execution of the method if we compare with the results base form without preprocessing. Actually, this result shows us there are no significant effects of this method for categorizing the rating stars of the reviews unlike the effects in messaging applications and social media as proved in some researches. Almost the same result for removing punctuations even after we execute multiple times for each method.

5.4.3 Classifier Performance Based on Expanding Abbreviations and Acronyms

In this part, we try to find some abbreviations and acronyms in the reviews of the restaurants but unfortunately, there is no remarkable number of the most common words of this specific field in the text of the reviews. Thus, we use general abbreviations and acronyms which are the most commonly used in messaging applications and social media. For example, omg (Oh my God), Lol (Laughing out Loud), 2moro (Tomorrow), B3 (Blah, Blah, Blah), etc. which we got from https://www.smart-words.org/abbreviations/text.html. Again, we use a replacer class for this purpose and choose space tokenizer for the review text in order to get direct simple effects of the method.

The accuracy results are 0.263 (KNN), 0.305 (DT), 0.393 (RF), 0.484 (LR), 0.470 (SGD), 0.498 (NB), 0.453 (SVM) after running the methods 20 times shuffled reviews. The average classifiers' accuracy results of expanding abbreviations and acronyms are not better than without execution of the method significantly if we compare to the results with the base form given in Table 5.4. Actually, this result shows us there are no significant effects of this method for categorizing the rating stars of the reviews unlike the effects in messaging applications and social media as proved in some researches.

5.4.4 Classifier Performance Based on Word Correction

In this part, we report the result obtained after executing the auto corrector of python on each review text for the misspelled words in order to change with their most possible similar words. As we mentioned above, misspelled words are checked according to the English language. We don't apply the removing Multiword Expression method due to a lack of a remarkable number of reviews as we observed from reviews text.

The accuracy results are 0.243 (KNN), 0.254 (DT), 0.321 (RF), 0.412 (LR), 0.393 (SGD), 0.421 (NB), 0.401 (SVM) after running the methods 20 times shuffled reviews. The average accuracy result of the word correction is much worse than other methods. As we observed from the output of the program, the tool which we used is not successful at all for the text of the reviews. Consequently, this result shows us there are no significant effects of this method for categorizing the rating stars of the reviews

unlike the effects in messaging applications and social media as proved in some researches.

5.4.5 Classifier Performance Based on Stopwords Elimination

In this part, we report the result obtained after removing the stopwords in each review text using nltk stopwords. As we mentioned above, stopwords are checked according to the English language.

Table 5.6 Classifier performance based on stopwords eliminations

Preprocessing Methods	KNN	DT	RF	LR	SGD	NB	SVM
Base Form	0.260	0.287	0.374	0.476	0.446	0.491	0.444
Removing Stopwords	0.261	0.376	0.389	0.494	0.454	0.512	0.462
Removing Stopwords with Modified List	0.299	0.293	0.406	0.494	0.461	0.523	0.486

As shown in Table 5.6 after running the methods 20 times shuffled reviews, the average accuracy results of the stopwords elimination are better than other methods. Especially the modified stopwords list method according to our text structure (like not removing comparative adverbs such as good, better, best) is slightly better than directly removing stopwords. As we observed from the output of the program after executing multiple times, we conclude that this method has a significant effect on categorizing the rating stars of the reviews.

5.4.6 Classifier Performance Based on Stemming

In this part, we investigate the stemming algorithms such as Porter Stemmer, Lancaster Stemmer, and Snowball Stemmer and their efficiencies on the restaurant reviews. We reduce words to root forms by removing prefixes and suffixes according to some grammatical rules of the nltk stemmers. We execute the stemmer algorithms on space tokenizer base form in order to get directly simple effects of them.

Table 5.7 Classifier performance based on stemming

Preprocessing Methods	KNN	DT	RF	LR	SGD	NB	SVM

Base Form	0.260	0.287	0.374	0.476	0.446	0.491	0.444
PorterStemmer()	0.248	0.297	0.376	0.512	0.459	0.511	0.463
LancesterStemmer()	0.232	0.297	0.373	0.501	0.453	0.498	0.448
SnowballStemmer	0.254	0.316	0.363	0.499	0.461	0.517	0.450

As shown in Table 5.7 after running the methods 20 times shuffled reviews, all the stemmer algorithms slightly change the average accuracy results, especially for Logistic Regression and Naive Bayes classifiers in a good way. In general, these results show us there are no significant effects of these methods for categorizing the rating stars of the reviews as we expected.

5.4.7 Classifier Performance Based on Lemmatization

This time we investigate the effects of the lemmatizer on the restaurant reviews. Again, we execute the lemmatizer algorithms on space tokenizer base form in order to get directly simple effects of them.

Table 5.8 Classifier performance based on lemmatization

Preprocessing Methods	KNN	DT	RF	LR	SGD	NB	SVM
Base Form	0.260	0.287	0.374	0.476	0.446	0.491	0.444
WordNetLemmatizer()	0.263	0.309	0.373	0.473	0.421	0.479	0.439
WordNetLemmatizer() with position	0.305	0.349	0.382	0.513	0.467	0.524	0.481

As shown in Table 5.8 after running the methods 20 times shuffled reviews, the lemmatizer without indicating the word position does not change the accuracy results significantly but the lemmatizer with position increases the accuracy results even we execute the program multiple times. As we observed from the output of the program, these results show us there are no significant effects of this method for categorizing the rating stars of the reviews as we expected but indicating the position of the word for the lemmatizer gives us better results.

5.4.8 Classifier Performance Based on Lowercasing

In this part, we report the result obtained after executing the lowercasing on each review text to increase the performance of the classifier without considering the non-

consistence of texts. We execute our lowercasing method on 20 times shuffled reviews and on space tokenizer base form in order to get directly simple effects of the method.
This time the accuracy results are 0.314 (KNN), 0.333 (DT), 0.399 (RF), 0.523 (LR), 0.473 (SGD), 0.522 (NB), 0.497 (SVM) and surprisingly the average accuracy result of the lowercasing method is much better than before if we compare to the results with the base form given Table 5.4. As we observed from the output of the program after executing multiple times, this result shows us there are significant effects of this method for categorizing the star ratings of the reviews. Because of treating each word as a dimension in the feature set, having the same words in different case cause the models to be confused and loss of time.

5.4.9 Classifier Performance Based on Removing Common Words

In this part, we report the result obtained after removing the common words on each review text to increase the performance of the classifier. We execute the algorithm on 20 times shuffled reviews and on space tokenizer base form in order to get directly simple effects of the method.
This time the accuracy results are 0.294 (KNN), 0.324 (DT), 0.374 (RF), 0.500 (LR), 0.471 (SGD), 0.518 (NB), 0.458 (SVM) and the average accuracy result of the removing common words method is much better than before if we compare to the results with the base form given Table 5.4. As we observed from the output of the program after executing multiple times, this result shows us there are significant effects of this method for categorizing the star ratings of the reviews. Because the classifier confuses the class of rating when seeing those common words in the review text.

5.4.10 Classifier Performance Based on Removing N-grams

In this part, we report the result obtained after executing some combination of Ngrams. In the beginning, we apply each Ngrams alone, and then we apply a combination of three in order to see the effect of each combination. Same as before, each obtained result is the average of 20 times shuffled restaurant reviews.

Table 5.9 Classifier performance based on removing N-grams

Ngrams	KNN	DT	RF	LR	SGD	NB	SVM

Unigram()	0.304	0.332	0.364	0.370	0.371	0.365	0.361
Bigram()	0.332	0.403	0.415	0.426	0.398	0.445	0.435
Trigram()	0.255	0.297	0.307	0.336	0.331	0.344	0.305
Unigram() & Trigram()	0.325	0.325	0.365	0.388	0.371	0.389	0.377
Unigram() & Bigram()	0.324	0.387	0.411	0.486	0.455	0.483	0.458
Bigram() & Trigram()	0.219	0.362	0.406	0.469	0.432	0.458	0.445
Unigram() & Bigram() & Trigram()	0.335	0.420	0.425	0.495	0.469	0.511	0.472

As shown in Table 5.9, we observed from the output of the program after executing multiple times, while the effect of the Bigram is bigger than Unigram, the effect of the Unigram is bigger than Trigram. When it comes to the combination of the Ngrams, the effect of the combination Unigram() & Bigram() is more than Bigram() & Trigram() while the effect of the Bigram() & Trigram() is more than Unigram() & Trigram(). We get the best result even after executing multiple times when we apply all the Ngrams together.

5.4.11 Classifier Performance Based on Preprocessing Order

In this part, we report the results obtained after executing some combination of preprocessing methods in order to see the effects of executing order. For this purpose, we use lemmatization, stopwords, and lowercasing preprocessing methods which have a positive effect on chosen classifiers on the review data set as we mentioned above. In order to see the difference between preprocessing orders we execute all the combinations of three methods, respectively. This time we don't shuffle the review set to see the effects of executing the order of three methods on the same dataset.

Table 5.10 Classifier performance based on preprocessing order

Order of the Methods	KNN	DT	RF	LR	SGD	NB	SVM
Base Form	0.260	0.287	0.374	0.476	0.446	0.491	0.444
Lemmatization – Stopwords - Lowercasing	0.345	0.394	0.439	0.553	0.512	0.564	0.521
Lemmatization – Lowercasing - Stopwords	0.358	0.398	0.423	0.535	0.514	0.547	0.524
Stopwords – Lemmatization -	0.360	0.376	0.423	0.567	0.522	0.567	0.536

Lowercasing							
Stopwords – Lowercasing - Lemmatization	0.344	0.371	0.448	0.542	0.512	0.570	0.527
Lowercasing – Stopwords – Lemmatization	0.339	0.404	0.445	0.547	0.494	0.561	0.514
Lowercasing – Lemmatization - Stopwords	0.344	0.391	0.433	0.545	0.507	0.542	0.515

As shown in Table 5.10, we observed from the output of the program after executing multiple times, executing the order of the preprocessing methods affects the accuracy results of any classifier by almost 2%. In addition, when we compare the accuracy results of each classifier with the base form, the preprocessing methods applied to change the accuracy results up to 10% in some classifiers. These results show us how important applying preprocessing methods are when classifying our data.

5.5 Conclusion

In this chapter, we discussed the experiments involving some simple text preprocessing methods that give an impact on the classification performance when we predict fine-grained review rating stars. For this reason, we wanted to show their effects on the five class-based review rating stars, individually.

Although less attention has been paid to the text preprocessing in the researches, our evaluations highlight that it has a remarkable impact on the performance of classifier but of course not for all the methods. Some of them have a positive effect on classification accuracy, while some have a negative effect, and others have a neutral effect.

In general, a simple stopwords elimination, lowercasing, removing common words, and lastly the combination of 1-to-3 Ngrams perform better than other preprocessing methods for improving the classification accuracy of the five class-based review rating stars. As we mentioned before, the challenge of this field is to predict fine-grained review rating stars because of being used almost the same words for the close classes. Otherwise, it might be useful to apply the mentioned methods, for instance, for the binary distinction of positive vs. negative. Namely the effects of the preprocessing

methods can change on any domain. So, it should be considered all possible preprocessing methods and their combination before used in any application. And applying the order of the preprocessing methods can also be important. The effects of abbreviations, acronyms, stemming and lemmatization might be higher after executing lowercasing to the text. It is believed that our study results will help future researchers to carefully select these text preprocessing methods.

6. CALCULATING OVERALL STAR RATINGS BASED ON REVIEWS

Through the instruments of increased e-commerce platforms, the customers' reviews and ratings have started to play a significant role in marketing strategies. Customers do not hesitate to share their negative or positive opinions on these platforms. Since most of the customers generally choose a product with at least 4 stars over 5 or 8 stars over 10, calculating the quality star rating of a product or a service has become extremely important. But, determining the star ratings of a multi-class rating system is quite hard, not only because of probability ratio but also used words which are very similar among the close classes. Hence, a binary classification is preferred. There are several ways to calculate the quality ratings according to the need, such us views of trustful users, usefulness of votes, number of the users providing ratings, date and time of the ratings, and the sentiment analysis, which is the method this paper is concerned with. We propose a methodology to overcome the challenges when calculating the quality of multi-class star ratings, specifically on restaurant reviews, to calculate the overall star ratings via sentence-based, review-based, dictionary-based, and the newly proposed hybrid-based sentiment analysis methods.

E-commerce can be broadly defined as the purchase and sale of a product or a service through the Internet and it is one of the most important building blocks in today's modern business world, as online shopping ranges from 13% to 82% in just European countries and B2C e-commerce expected to rise to €602 billion just in 2018 [103]. Namely, the importance of e-commerce is growing day by day. Thus, almost all types of businesses start to find out the most effective ways to influence the customers for their e-commerce market. Online ratings and review websites are some of the most powerful ways to changing purchasing decisions and increasing customer confidence. Especially it has become vitally important for small businesses to fight with conglomerates that dominate the related market.

Ratings and reviews are not just for customers to find good quality products and trustful sellers but also for the product producer to identify the pros and cons related to their products and to determine competitive intelligence. But due to the charming market share, countless misleading reviews and ratings are showing up with each passing day

by malicious users, biased bloggers, even owners of the related products. Some systems try to solve this problem by helpfulness votes, but it is found that the ratings for the most helpful reviews are consistently inflated compared with the ratings provided [104]. Of course, today's product ratings and reviews systems are very useful for the purchasing decisions of the customers but there are still weaknesses, particularly the difference between the product ratings and reviews text [105].

For an efficient and productive online shopping, not only there should be a secure system, easy to understand content, prompt delivery and quality services, business credibility, etc. [106], but also a high-quality star rating calculation in order to increase customer confidence since more and more customers rely on the opinions of other users when making a purchasing decision. Thus, to get a more reliable and comprehensive rating system for products, a few methods such as views of trustful users, usefulness of votes, number of the users providing ratings, date and time of the ratings, and the sentiment analysis are needed. In this way, the effects of the fake accounts on the rating systems can be lowered significantly or eliminated. Amongst above mentioned methods, we focus on the sentiment analysis of reviews, which is one of the ways to check the quality of star ratings of products and services. To this end, we compare sentiment analysis methods and develop a hybrid model to classify each review text.

There are a lot of existing work on similar topics of various scopes, most of which focused on the binary distinction of positive vs. negative but our model predicts the user's numerical star ratings in a Likert scale, which is the main challenge to determine the exact star among close classes. After training model on the feature set, we calculate each restaurant's star rating from users' reviews by a hybrid model consisted of supervised learning (document level and sentence level) and dictionary-based approach and then analyze the results by comparing to the real ratings to see the pros and cons of each different approach.

The rest of this chapter is organized as follows. After the introduction, Section 6.1 presents some of the recent work especially the research on review rating prediction using sentiment analysis of the reviews. In Section 6.2, details about research methods and some specific examples about the related field are provided. Section 6.3 introduces a real-life dataset used in our experiment and Section 6.4 reports some experimental outcomes and evaluates the results. Finally, we conclude and discuss in Section 6.5.

6.1 Related Work

Although advertising still has a substantial effect on the increased product sales, sentiment analysis of product reviews and ratings has attracted a great deal of attention in recent years because of providing high-profit share in e-commerce platforms. In this section, we specifically focus on some recent related research, which aim to put into practice binary or fine-grained classification and latent aspect rating. So, the research that is close to the content of our research is summarized below.

Govindarajan [107] proposes a hybrid model consisted of Naïve Bayes, Genetic Algorithm, and Support Vector Machine (SVM) in order to indicate the effects on the restaurant reviews with a comparative analysis. For this purpose, classification accuracy was evaluated by each model individually first, and then by the ensembled ones. According to their results, the recommended hybrid model delivers better performance than the base classifiers on the restaurant reviews with regard to classification accuracy.

Guo et al. [108] investigate two types of estimations, the first one is the star rating of restaurants and the second one is the popularity change of restaurants. They use features such as price, location, available services, etc. by using a couple of machine learning methods such as Logistic Regression (LR), Naïve Bayes, Neural Network and SVM. However, according to their results, actually, none of the methods tried gives good results due to lack of relevant data.

Yu et al. [109] compare the performance of some machine learning algorithms in order to predict star ratings of reviews related to restaurants. They use linear regression, random forest, and the latent factor model. According Yu et al., the Random Forest (RF) is the best model for predicting the rating of reviews because of using reasonable features extracted from the rich dataset.

Asghar [110] tries to find out the best model for predicting the star rating of reviews, as a five-class classification problem, among sixteen different models combining four feature methods like 1-to-3 N-grams and Latent Semantic Indexing. According to the results of four machine learning algorithms, LR seems the best one, obtained by unigram & bigram on the set of the top 10.000 features with the accuracy of 64%, among the others such as Naïve Bayes classification, Perceptrons, and Linear Support Vector Machine classification.

Kapukaranov and Nakov [111] focus on comparing classification, regression, and ordinal regression in terms of performance on the reviews text and contextual features such as movie length, director, actors, etc. According to their results, regression and contextual features model is better than the other combinations. While country, directors, or genre, namely factual information, except actors do not seem to be useful, user average score is the most useful contextual features. As a result of the research, we now know that adding contextual information has a positive impact on performance.

Ghazvinian [112] tries to categorize numerical ratings of multi-class restaurant reviews rather than to predict them simply positive or negative. He implements a maximum entropy classifier with a selected feature set (unigram, preprocessing, bigram, etc.) and the sentiment models such as language models and sentiment modeling. According to the results, the selected/tried model after a couple of experimental results can predict the rating of a review at 60% precision.

Lee et al. [113] focus on the evaluation of the user-generated and machine-generated star ratings using Naïve Bayes and SVM. After applying some preprocessing methods on the obtained data and then vectorizing it, they calculate star ratings of reviews using sentiment analysis. According to the results, the combination of the VADER-Sentiment Analysis tool to produce star value and sending directly vectorized data to the train is the most viable approach.

Doan and Kalita [114] study on an incremental learning approach by a modified online RF model in order to overcome retraining the whole system problem whenever new data become available, namely streaming data. According to the experimental results after a couple of data processes, the proposed approach comes in third place in five different models following by Factorization Machine and Hoeffding Tree, but it needs longer run time because of high computation involved.

Zhang et al. [115] suggest a model for Yelp Dataset to predict reviews' usefulness and examine three well-known classification models: K-Nearest Neighbors (kNN), SVM, and RF Classification. To improve the success of the model, they choose some users, reviews, and business-related features and evaluate their performance using LR. According to the experimental results after selecting features, RF without TF-IDF features gives the best results with the accuracy 0.699.

Huang and Yu [116] suggest a novel task to restore the truthful rating in order to overcome fake reviews problems that mislead not only users but also service providers. For this purpose, they evaluate the performance of a couple of models such as Linear Regression, kNN, Deep Neural Network (DNN), etc. on the two weeks summarized ratings. Especially DNN with some sub-models for different features at a specific layer and full connection in the following layer gives the best performance among all the other methods.

In order to increase rating prediction accuracy, Ochi et al. [117] work on a novel feature vector, dimension of which is reduced using extracted feature words in order to execute algorithms on a large and sparse dataset practically. For this purpose, they create a feature set finding words, which is appeared in the reviews too many times to increase the density of the matrix. According to their results, the studied approach improves the prediction accuracy on a corpus of golf reviews.

Jin et al. [118] suggest a model for review rating prediction to improve the accuracy by obtaining semantics of review text and completing the value of missing ratings looking at the history of the user's behavior applying on two different datasets. According to the experimental results, using the skip-thought vector of review text and filling missing ratings improve the prediction accuracy more than that of the other combined methods.

Since the reviews are written for more than one field about a product or service, Wang et al. [119] suggest a new method based on their previous work called LARA which generates rating on a set of predefined aspects and relative weights placed by a reviewer on each aspect. In contrast with the previous method, the new method called LARAM doesn't need to specify aspect keywords by users. According to their results, LARAM can effectively find latent topical aspects, ratings on each identified aspect, and weights placed on different aspects when generating the overall rating.

Xu et al. [120] propose a model to predict hidden aspect ratings such as "cleanliness", "food", "service", etc. from the users' reviews. Actually, the proposed model is compared with LARAM [119] because of having the same concept. According to their results, it can also alleviate the aspect sparsity issue, where it is claimed that LARAM cannot effectively handle. Experimental results show that the predicted aspect ratings for each review is more accurate and reliable based on the proposed model.

Xu et al. [121] compare the performance of different learning algorithms in order to predict reviews ratings received from the Yelp Dataset. After executing several preprocessing on the data, they use an existing opinion lexicon and build a feature dictionary to evaluate the algorithms based on precision and recall. According to the results, binarized Naïve Bayes is more effective than both Perceptron and Multiclass SVM algorithms to predict star ratings.

Chen et al. [122] use long short-term memory (LSTM), which is a part of deep learning method, to fine-grained sentiment analysis on Chinese phone reviews. The study also compares the algorithm with polarity analysis on Chinese and English texts by accuracy and F-score. Even though the method is applied to Chinese text, the result shows that the application of the LSTM on fine-grained sentiment analysis is effective after a series of modifications.

Chauhan et al. [123] focus on the polarity sentiment analysis in order to detect whether electronic product reviews are fake but not by calculating the words' weight from the created dictionary. After calculating the sentiment score of the reviews by using NLTK and VADER with a set of discriminative rules, the proposed model shows effective results.

Kumari et al. [124] investigate the polarity sentiment analysis in order to determine whether the smartphone product reviews are positive or negative using SVM. Obtained performance results by using the values of precision, recall and f-measure, and accuracy, the proposed SVM work performs very effective, robust, and better than those other methods compared.

Barbosa et al. [125] study on sentiment analysis to determine overall ratings of hotels from the review text by comparing three different algorithms: OpinionFinder, Stanford CoreNLP, and the Naïve Bayes combined with sentiment lexicon. Actually, the authors try to understand whether reviews are correlated with overall rating, namely whether the reviews' texts are reliable or not. Consequently, the results show that reviews are correlated with overall ratings and they can be used for predicting numerical ratings.

Yang and Chao [126] focus on the sentiment annotation to highlight the effectiveness of their approach on the overloading information about tourism reviews in Chinese at the sentence level. Even though it is applied on a limited corpus the proposed approach shows that adding sentiment annotation at sentence level improves the information

quality of the original review, namely it makes them sufficient, concise information, and more understandable.

Vinodhini and Chandrasekaran [127] evaluate the performance of the neural network based on sentiment classification methods with five quality measures and two statistical methods which are Support Vector Machine and Linear Discriminative Analysis on online reviews. According to their experimental results, feature reduction is important for learning methods and homogenous ensemble methods give better results than other classification methods.

Fang and Zhan [128] elaborate on the problem of sentiment analysis especially on focusing sentence-level and review-level categorization experimenting on a set of online product reviews. Both categories' results are promising that average sentiment scores are satisfied enough for polarity categorization.

Malik et al. [129] evaluate reviews when calculating the overall sentiment obtained from an e-commerce website. They estimate the opinion polarity using the weight method but for particular features of a product entered by a customer. Namely, they prove that the proposed model depending on the priority of user wishes works more effectively.

As mentioned above, while most of the studies focus on the performance of different classifiers in sentiment analysis, some of them examine the effects of the properties in the dataset on the performance of classifiers. Another part of the studies investigates the effect of sentiment analysis in different corpus. According to the results obtained from the researches, different classifiers can perform differently on different data set. In addition, we see that the effect of each feature in the data set varies according to the relevant field and preprocessing methods also affect the performance of the classifiers.

As it is mentioned, the effect of product reviews on e-commerce platforms is known. The fact that there are not enough studies on data sets that contain numerical star ratings in a Likert scale, and that the product reviews consist of short text and that these texts contain similarities in close classes, bring difficulties in terms of scoring the product.

In this context, apart from the performance of the classifiers that are frequently investigated in the relevant field, we are also looking for better results using different sentiment analysis methods. In addition, the success of ready-made libraries used in this field is compared with related methods.

6.2 Prepare Background and Context

Nowadays, almost no one buys any online product without reading other users' comments and experiences since user-generated content has become the mainstream of the e-commerce platforms. Actually, it would be wrong to limit it just with the e-commerce platforms, experiences in our daily lives are also important, sometimes crucial. Even for some basic things, we need advice such as when we want to watch a movie, listen to music, or read a column. In fact, this situation is due to the limited time and internet pollution created by the modern world. Of course, people intrinsically want to reach the highest quality content as soon as possible. At the same time, it is an opportunity to be informed in the related field in a limited time, and having a grasp of concept helps to measure the quality and price conditions without any effort related to the service to be received or to be offered.

On the other hand, reviews make a dynamic content for the related e-commerce platforms. A wealth of worthwhile customer feedback can provide electronic word-of-mouth (e-WOM). Although companies generally produce original contents, since they identify with the corporate language of the brand very much, they produce generic content without realizing it, but when the customers make an evaluation, they make both emotions and original reviews, which are the most valuable facts in marketing efforts and make remarkable influences on purchasing decisions of other customers.

Increasing the importance of reviews and ratings in e-commerce platforms, malicious users and companies that want to increase their profit margin have come up with indirect ways such as creating fake accounts or voting just positive reviews to influence such systems and increase their products' ratings since a lot of recent researches show that almost over the 90% of consumers read online reviews and 88% of them trust the online reviews and ratings as much as personal advice [130]. Likewise, 57% of consumers do not want to use a business with less than 4 stars [131]. Thus, rating algorithms have tried to find new methods to rule out fake accounts or reduce their impact on the star rating calculation such us based on trustful users, usefulness votes, the number of the rated users, time, sentiment analysis, etc., according to the platform used. In this research, we focus on sentiment analysis on product reviews in order to classify multi-class review ratings more effectively. As we mentioned above, it is a challenge that classifies multi-class review ratings because of the similar words used in

closed classes. For this purpose, we try to create a hybrid model consisted of supervised learning (document level and sentence level) and a dictionary-based approach.

6.2.1 Sentiment Analysis

Sentiment analysis is a field that focuses on specifying people feelings (such as being happy, surprised, angry, funny, sad, pleased, satisfied, etc.), opinions (such as thinking about a product that it is useless, good quality, waste of time, too expensive, worth a try, etc.) or sentiments towards a situation, entity or event, specifically a product or service for our research by looking at a piece of writing or document. In fact, as can be seen in recent research it is not restricted just with writings but also can be specified by other structures such as emoticons, emojis, etc [132]. Sentiment analysis in terms, our research refer to evaluating reviews about a product or service sometimes to classifying them whether they are positive, negative, or neutral, sometimes grading them within a certain range. Actually, sentiment analysis can be used not only for classifying of opinions but also summarizing the main subject of a document, determining whether a sentence is subjective, finding related document for a given query or for other reasons such as determining types of the writing, owner of the writing, spam texts, etc. Sentiment analysis used for examining review ratings in our research by comparing it with star ratings. In Figure 6.1, you can see how to sentiment analysis can be branched according to the approach used.

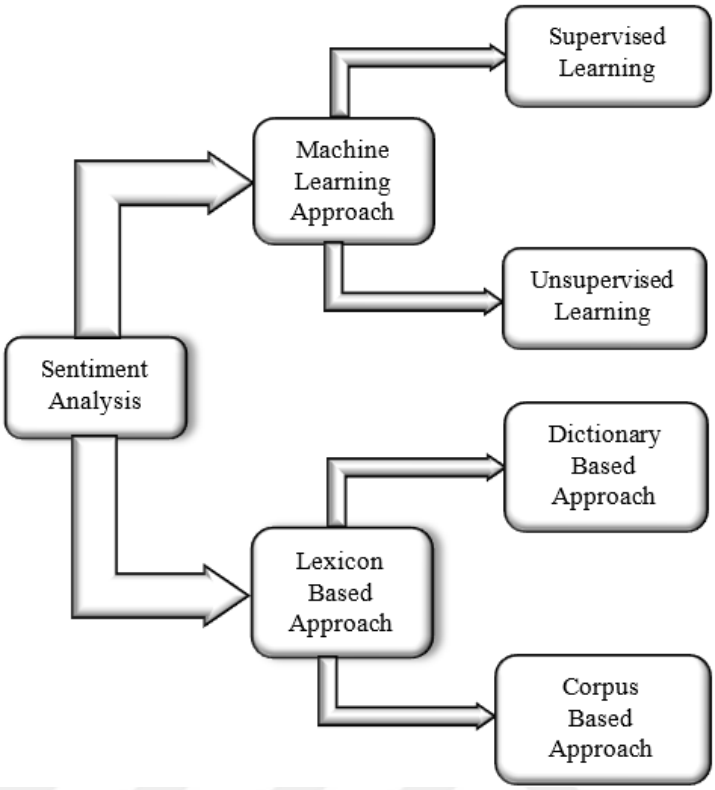


Figure 6.1 Sentiment analysis techniques [124]

6.2.1.1 Machine Learning Approach

Machine learning, which is based on artificial intelligence, is a learning method that provides the most appropriate response to the new situation obtained by its past experiences. In this context, the characterized, sized and well-prepared data set directly affect the quality of the response to the new situation. So, making a good prediction in a completely new situation is its weakest point. For the sentiment analysis, it uses this logic to classify an opinion of a text with the help of linguistic features.

In supervised learning, each record, in our case reviews, in a given data set is labeled including a specific class. Then, for an unknown class instance, this model uses past experiences which are the given data to make a tag to predict the new instance. In our case, tagging a new review as a positive, negative or neutral, or giving a star rating out of 5.

In unsupervised learning, data has no class information. The main purpose is to extract information from this data by clustering it based on a relationship between variables. In our case, clustering reviews by similarity (in terms of meaning) or by their usage together.

6.2.1.2 Lexicon Based Approach

This approach works by processing the data generated according to the structure of a language. The strongest side of this approach is that it does not require any labeled training data. Namely, for categorizing the text it calculates the polarization (it can be negative, positive, neutral, or numeric scores) of the words that convey feelings used in the text according to the pre-prepared domain dictionary. Then, to get the overall sentiment of the text based on that sentiment polarity calculation it uses some classification algorithms. Consequently, the lexicon-based approach is a collection of words, in other words, it is a sort of pattern matching. There are a lot of popular lexicons such as SentiWordNet, SentiStrength, OpinionFinder lexicon, AFINN lexicon (emotional ratings), NCR lexicon, etc. There are two general approaches in the lexicon-based approach. The first one is the manual approach which is really hard to generate it with all aspects, namely, it takes too much time to fix it and requires some experts corresponding to the studied field. The second one is the automated approach which consists of dictionary-based and corpus-based.

In the dictionary-based approach, the words related to general-purpose fields are determined, which start with a smaller set called seed words collected manually in a dictionary. Then this small set is iteratively expanded by adding synonyms, antonyms, etc., finding with the help of a dictionary-like WordNet until a new word cannot be found [133]. After that, the model compares the text with the dictionary in order to determine the polarity degree. The key point is to convert all the words of the text according to the prepared dictionary form otherwise the words in the text can't match with the words in that dictionary. Of course, this process is not easy for all languages in which words' roots often change and become unrecognizable when deriving new words and sometimes words also have a complex suffix and prefix structures as in Arabic, French, Farsi, etc.

The corpus-based approach can be particularly used for domain-specific applications and words are collected in the same way as the dictionary-based approach, but the collected words are polarized according to the field. Namely, while some words are thought to have a positive polarity in some specific domain, they can have a negative polarity for some others. For example, for the word "long", saying that "It was a too slow and too long movie" for a movie domain can be carried a negative meaning,

"Taking a long position here" for a financial domain can be carried positive meaning or "that's a long noodle" for a restaurant domain can be carried neutral meaning. Namely, the meaning of a word can be different in different situations or domains [134].

6.2.2 Data Preprocessing

In our research, we use the dictionary-based approach and supervised sentiment learning under the branch of the Machine Learning approach. The following section presents the methods and classifiers used in the study as shown in the flow diagram in Figure 6.2.

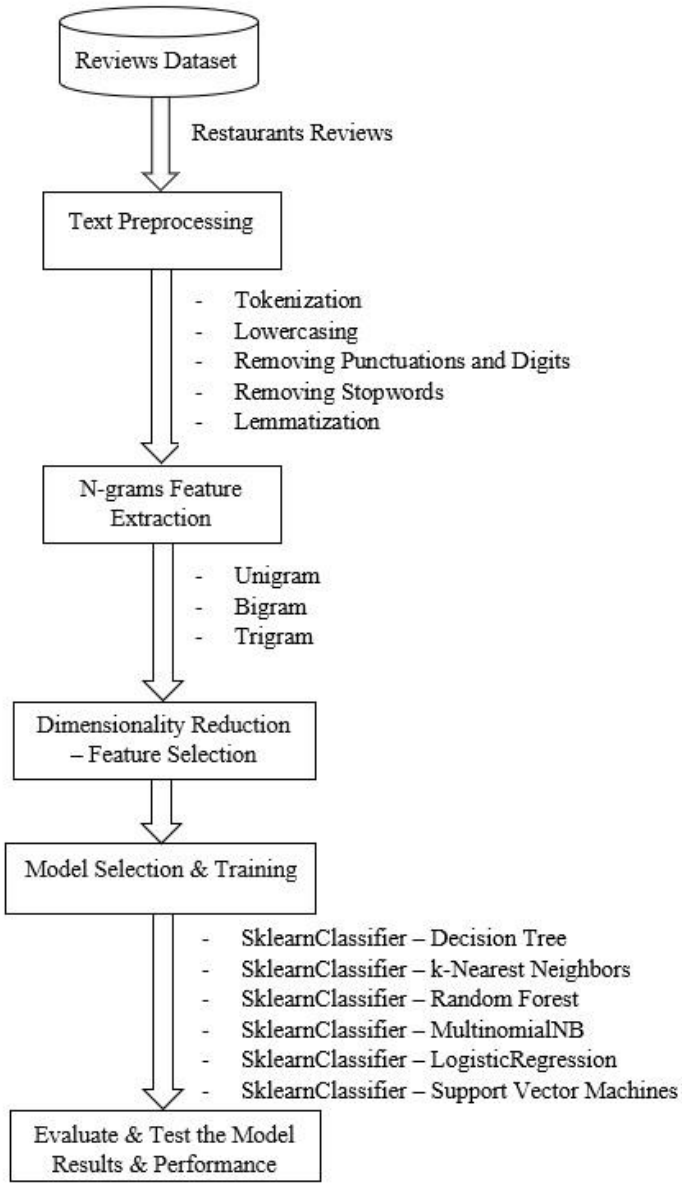


Figure 6.2 Flow of the proposed model

6.2.2.1 Tekonization

In order to get as an input in Natural Language Processing (NLP) processes, a review text must be divided into the smallest meaning units called tokens such as words, phrases, symbols, or sometimes whole sentences. For this purpose, generally several Natural Language Toolkit (NLTK) are used according to the needs of the field studied. Some of the NLTK tokenizers are space tokenizer which breaks up a sequence of strings using space between two tokens, TreebankWord tokenizer which also treats punctuations as a token, and Regexp tokenizer which breaks up the text using regular expressions. In our research, we use Regexp tokenizer.

6.2.2.2 Lowercasing

In order to prevent case sensitivity in sentiment analysis, we need to convert all text into the lowercase or vice versa. In this way, when we create our feature set, our algorithm does not treat the same words which are in a different form as a different dimension. Actually, this preprocessing method has a great effect on the accuracy of classification as we experienced in chapter 5.

6.2.2.3 Removing Punctuation and Digits

As in lowercasing, when the tokenizer split up a review text into tokens, it treats punctuations and digits as sperate tokens which increase the feature dimensions and are useless as it can be seen in most of the sentiment analysis studies. We also remove all punctuations in our feature set which don't have any significant effect on the accuracy of the classification.

6.2.2.4 Removing Stopwords

Removing stopwords is another important step of preprocessing and it has a really great effect on the accuracy result as we explained the reason for dimensions. Namely, since the stopwords increase the dimensions of the feature set and there are numerous of them in texts, it is better to get rid of them. As it is known, we define stopwords as the most common words found in a language. So, having stopwords in the feature set is not a

distinctive feature to catch the difference between classes. If we give some examples to the stopwords, we can list words like “a”, “the”, “an”, “me”, “your”, “will”, etc. Of course, removing stopwords can be changed according to the application. Namely, while “once”, “when”, “since”, etc., can be important words for some application, “will”, “can”, “do”, etc. can be important for some others [135]. According to the application, researchers can change the stopwords list.

6.2.2.5 Lemmatization

One another important method is lemmatization. Actually, while some researchers use stemming which tries to find out the root form on the words by removing suffix and prefix according to the related language grammatical rules, some of them use lemmatization which finds dictionary forms of the words categorized them as a part of the speech. Namely, lemmatization reduces words according to the position in a sentence. In this way, we find a base structure of all words in a review. Otherwise, it is meaningless to store all versions of a word which have the same meaning such as “go”, “went”, “gone” or “interesting”, “interested”, “interest”, “interests” due to the same reason that all these versions are considered as a unique feature [136].

6.2.3 N-grams Feature Extraction

In order to identify discriminative and useful features from the text of the reviews, we should determine the words which represent their own related category. For this purpose, we create a feature vector called indexing which is generally used by Information Retrieval. Of course, there are other methods to represent textual data, but the feature vector is one of the easiest structures. In this method, for each different word passing in the text of the reviews, a separated dimension is created. Because of this, a matrix is created which shows which review has which words and how many times. It is called a bag of words that is used for training a classifier. So, the matrix consisted of vectors or arrays would be quite large because of containing all elements for each possible word and for the each occurred word in the related review text weight is calculated by the function tf-idf (term frequency-inverse document frequency) measure that calculates how important a word in a text document. While TF calculates how often

a word occurs in a review text by dividing a total number of words in the review text, IDF calculates a logarithm that measures a total number of the reviews dividing by the number of reviews which have that specific word [137]. Namely, in IDF, if a word occurs in almost all the reviews text, it means that that word is not a discriminative word for classification, namely less importance is given to common words. You can see an instance vector space model from Table 6.1.

Table 6.1 Tf-Idf vector space model

	Review_1	Review_2	Review_3	Review_4	Review_5	…	Review_n
Word_1	8	0	1	0	2	…	3
Word_2	0	2	4	0	1	…	0
Word_3	2	0	0	1	0	…	3
Word_4	5	0	0	0	1	…	0
Word_n	1	2	1	2	0	…	4

One another important point about text representation is the N-grams method. It has a very great effect on the performance of the classifier, especially for categorizing the product review. Because sometimes combining of the words can represent better than handling each word separately. For instance, a review that contains phrases such as “Meals weren’t delicious in that restaurant”. If we get the words individually, because of the words “delicious” which has a positive meaning, a classifier can classify as a positive review but if we get the text as a combination of words with one before and one after word (“weren’t delicious”), the classifier can easily find out that is a negative review [138]. So, it gives unsurprisingly better results because of capturing relationships between occurred words in the text.

In N-grams, the number of the “N” can be changed according to the application field. After observing many experimental results, we decide to combine unigrams, bigrams, and trigrams. Although the overwhelming supremacy of the bigrams, the effect of combining all three types of N-grams is slightly more than other N-grams combinations on the review text. Even the same trigrams rarely occur in different review text, it can make the classifier’s job easier.

6.2.4 Dimensionality Reduction

In order to increase the performance of the classifier, we also reduce the dimensions by removing some features that occurred in some review categories at the same time. Firstly, we remove some words/N-grams phrases which occur in all categories and are not discriminative features anymore. After that, we remove some other specific words/N-grams phrases that belong to the closed classes. Because while the classifier can determine the exact class between distant classes easily such as between 1 to 5 or 2 to 5, it has difficulties in distinguishing close classes such as between 1 to 2 or 4 to 5. We came to this conclusion as a result of some experiments we repeated many times and we got better results. After this last pre-processing, our data is ready for classification models.

6.2.5 Existing Classification Methods

Since the past decades, researchers have focused on designing a better model that provides the right predictions to classify a piece of text written by people looking at the data at hand. This is quite important because analyzing people's thoughts from what they write can give strong clues for their next behaviors, and this is a piece of vital information for especially e-commerce platforms. Nowadays hundreds of applications using this logic such as in search engines, spam detection, speech recognition, fraud detection, advertisements, etc.

Here we use some machine learning models to analyze the text of the reviews to classify them for better recommendations by all types of e-commerce platforms or by those that are basing their customers to digital platforms. Our ultimate goal is to compare the sentiment analysis methods whether the star ratings match the comments in general and create a hybrid model to get better results. For this purpose, we use a couple of models to see which one gives better results for our aims, some of them are Decision Tree (DT), k-Nearest Neighbors, Random Forest, Multinomial Naïve Bayes (MNB), Logistic Regression and Support Vector Machines.

6.2.5.1 Decision Tree

Decision Tree is one of the most preferred machine learning algorithms due to the fact that the logic of it is easily understood by people and gives good results. It is used for both classification and regression problems. The decision tree creates a training model over a data set that contains a large number of records and try to divide this data set into smaller subsets with a set of rules that it creates from this model. Thus, with these decision rules, it tries to determine which subset it is in, starting from the root of the tree for a new record. Then, at each iteration, the value of the new incoming record is compared to the value in the next internal node, and according to the comparison result, this process goes up to the leaves of the tree, thus the location of new incoming record is determined [139].

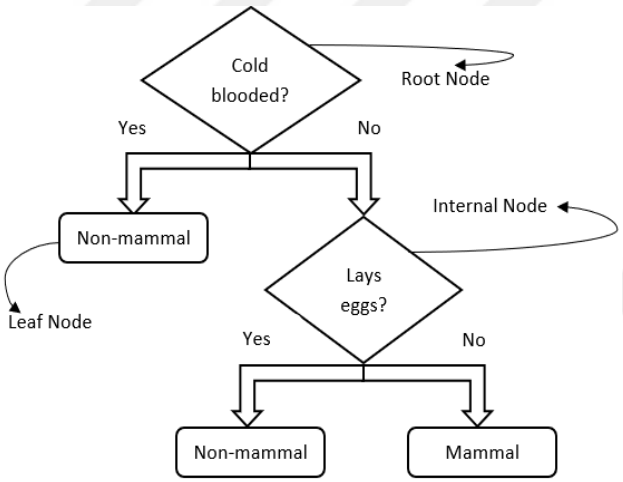


Figure 6.3 A simple Decision Tree

As is seen in Figure 6.3, the root node represents the entire population, except the root node, if a node can split into further sub-nodes, it can be called an internal node, otherwise, it is called a leaf or terminal node. The most important point when forming a decision tree is to divide it into homogeneous sub-sets as much as possible. For this purpose, various algorithms such as Iterative Dichotomiser 3 (ID3), Classification and Regression Tree (CART), CHi-squared Automatic Interaction Detector (CHAID) are selected.

6.2.5.2 K-Nearest Neighbors

The k-Nearest Neighbors algorithm is one of the supervised learning methods and it is an algorithm used in the solution of both classification and regression problems. The algorithm tries to find the class of newly arrived data by comparing it with the examples in the training set. For this purpose, it uses similarity measures such as Euclidean, Manhattan, Minkowski, Hamming, etc. It is a lazy learning technique as it memorizes the data set rather than learning the training data for classification. That is, when algorithm classifies a new data, it returns to the raw data each time to find the closest neighbors in the data set [140]. We can say that the basic working principle of the kNN consists of the following stages:

- "k" value, indicating the number of neighbors to be selected, is determined.
- With any similarity measurement, the distance of new data to other data is measured.
- The distances are listed, and the closest neighbors are found according to the "k" value.
- The number of each separate category is determined.
- The most found category determines the class of new data.

One of the important points in the algorithm is to determine the "k" value. For this purpose, an optimum "k" value can be determined by testing various "k" values on the training set. Otherwise, if the value of "k" is determined to be greater than the optimum value, it will cause an increase in dissimilar categories, and a small determination of the "k" value will cause the probable real class of the new data not to be found.

Another important point is the similarity measured used. Euclidean distance, one of the most used similarity measurements in this field, is the square root of the sum of squared differences between corresponding attributes of the two data [11]. With given $x_1 = (x_{11}, x_{12} \dots x_{1n})$ and $x_2 = (x_{21}, x_{22} \dots x_{2n})$,

$$dist_{Euclidean}(x_1 x_2) = \sqrt{\sum_{i=1}^{n} (x_{1i} - x_{2i})^2} \tag{6.1}$$

6.2.5.3 Random Forest

Although the basic operating structure is similar to Decision Trees, the Random Forest algorithm instead of a single decision tree, it randomly divides the data set into multiple subsets and creates a separate decision tree for each subset that operates as an ensemble. Then, if our problem is a regression, the algorithm gets the average of the results, and if it is a classification problem, this time the one with the most votes is chosen. This means that the same algorithm can be used for both classification and regression.

Besides this, the random forest algorithm gives more accurate results as the number of trees in the generated forest increases. Therefore, the overfitting problem, which is one of the missing points of decision trees, is overcome to some extent [141].

The most important point in this algorithm is that the most important feature is not to be searched when splitting a node like in decision trees. Instead of this, a subset of all the features is considered for splitting each node in each decision tree created by the algorithm.

6.2.5.4 Multinomial Naïve Bayes

The Multinomial Naïve Bayes is actually a type of specialized version of Naïve Bayes and it is generally used on multinomially distributed data for more text categorization. Namely, when the simple Naïve Bayes checks that particular words are included in the related review text or not (binary check), MNB explicitly takes into account that how many times those words occur in the related review text [142]. So, in the feature vector, each dimension represents the number of occurrences of each word in a review text. To predict the class of a review we use (6.3):

$$P(c) = \frac{N_c}{N}\ , \quad P(w\backslash c) = \frac{count\ (w,c)}{count(c)+|V|} \tag{6.2}$$

Where,

-P(c) represents the probability of a class,

-N_c represents the number of reviews in that class,

-N represents the total number of the reviews the dataset,

-P(w\c) represents the likelihood of a word given a class,

-Count(w, c) represents the count of the word occurring in that class,
-Count(c) represents the count of all words in that class, and
-|V| refers to the total number of unique words in reviews text.

6.2.5.5 Logistic Regression

Logistic Regression is another popular and widely used models for classification problems. The emergence of Logistic Regression is due to some shortcomings in Linear Regression, particularly when some outlier samples change the decision boundaries in Linear Regression especially in classification problems [143]. We can say that Logistic Regression provides the probability of a certain class, namely, predict only possible discrete outcomes but Linear Regression's outcomes can be any continuous values. For our research, the Logistic Regression algorithm uses the words and ratings of the reviews from the feature vector to create a model to predict the class of a given review. The logic of this algorithm is based on the prediction of two possible outcomes (Binary Logistic Regression), but also for the multi-class classification problems it uses the same logic. That is to say, firstly it gets the first class as a positive class and gets the rest as a negative class like doing it binary classification, and then it trains the classifier on this training set and gets the result. In the same way, it calculates for the other classes. In the end, it decides the class of example according to the obtained results. As a result, it calculates the highest probability computing for all class labels (for us star ratings from 1 to 5). The general equation of this algorithm is as follow:

$$P(y_i = 1|x) = \frac{1}{1 + exp^{-(\theta^T x)}} \quad (6.3)$$

$P(y_i = 1|x)$ is the probability of a class $y_i \in \{0, 1\}$ be in class 1 given the set of feature vector x_i.

6.2.5.6 Support Vector Machines

Support Vector Machines are supervised machine learning models used to solve classification and regression problems. The SVM is often used to classify linear data as well as nonlinear data. The basic logic of this algorithm is to determine the best line for separating points that consisting of different classes placed on a plane. Multiple lines

can be drawn to separate points of different classes. But it tries to find an optimum hyperplane to correctly classify each newly arrived data. For this purpose, it tries to find the plane that separates the points of each class by maximum distance that has the maximum margin [144].

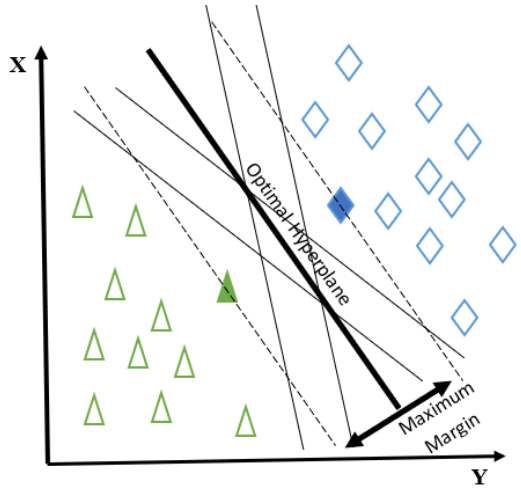


Figure 6.4 Optimal and Possible Hyperplanes

As it is seen Figure 4, support vectors are data points that interior are colored. These points are close to the hyperplane and can be more than one of course and each newly added point can change the location of the hyperplane. This can also happen if points are deleted. According to these determined support vectors, we create our SVM algorithm. Namely, once the model parameters are determined, the algorithm uses only on a subset of these support vectors to estimate the class of the new incoming data [145]. Therefore, support vectors define the margins of the hyperplane. As a result, each new data that comes in is checked whether it is below or above the hyperplane and then classified.

6.3 Dataset Description

In this study, the dataset was gathered from Yelp (https://www.yelp.com/dataset/) in June 2018 to evaluate the proposed model. It is a real e-commerce dataset that contains information about businesses such as business attributes, reviews, users, check-ins, tips, and photos in a five-separated file, which is suitable for many kinds of mining tasks. But specifically, we focus on two files of the dataset related to our research. The first one is the reviews file we get texts from to work on and the second one is a business file that we pull out just the restaurant business in order to get more specific results.

Table 6.2 Appearance of the review dataset

Business_id	Date	Review_id	Star	Text	User_id
6oRAC…	2012-06-14	IESLB…	4	I have no idea why some people give bad review...	0a2Ky…

The Review dataset is a json object in the review.json file, which specifies "User_id" who wrote the review, "Busines_Id" which the review text is written for, "Star" is a numeric rating out of 5 given by the user, "Date" shows the date of a review written for the related business, "Review_id" and the "Text" which we use for sentiment analysis. There are 229.907 reviews in which 43873 users have at least one review. Table II shows that the user 0a2Ky… has a review with id IESLB… and gives 4 stars for the business 6oRAC… on the date 2012-06-14. It is a 206.0 Mb file.

Table 6.3 Appearance of the business dataset

Business id	Categories	Address	Name	Review count	Star
PzOqR…	[Food, Bagels, Delis, Restaurant]	6520 W Happy Valley Rd\nSte...	Hot Bagels & Deli	14	3.5

The Business dataset includes some attributes such as categories which are just the business that included restaurants, full business address, Business name, Review count which shows how many users wrote a review to the related restaurant and its average star. There are 11.537 businesses which have at last one review. Table III shows the business PzOqR, which reviewed by 14 people and gets 3.5 stars on average and it is in the Food, Bagels, Delis, Restaurant category. It is a .json file and 4.08 Mb.
Our algorithms are executed on the Jupyter Notebook with Python version 2.7.11.

6.4 Empirical Observation

In this section, we carry out several experiments on our preprocessed data in order to verify the effects of learning algorithms on the performance of the classifier. According

to the results, we evaluate the sentiment analysis methods via comparing star ratings obtained from reviews text with real star ratings of the related restaurants.

As we mentioned above, we aim to compare review-based (RB), sentence-based (SB), and dictionary-based (DB) sentiment analysis with hybrid-based (HB) sentiment analysis which we created by the combination of these three methods and try to find the best performing method. For this purpose, we use Decision Tree, k-Nearest Neighbors, Random Forest, Multinomial Naïve Bayes, Logistic Regression and Support Vector Machines classifiers to get accuracy results for all types of sentiment analysis methods.

In order to implement our methods and to create our feature set, we extract 10000 reviews from the database that contain only restaurant reviews. As shown in Figure 6.2, these 10000 reviews from the database are arranged in a specific form in order to avoid unnecessary dimensions, such as Tokenization, Lowercasing, Removing Punctuations and Digits, Removing Stopwords, Lemmatization, and to apply our methods more efficiently. Then, in order to identify discriminative and useful features from the text of the reviews, we apply N-grams feature extraction combining with unigrams, bigrams, and trigrams. As mentioned before, the biggest challenge is to find the star rating of close categories because of using the same words. For this purpose, we reduce the dimensions by removing some features (the same N-grams) that occurred in some review categories at the same time to increase the performance of the classifiers.

In order to see the performance of each classifier over the created feature set, we selected 200 reviews from each category. In this way, we have a chance to analyze the classifier performance over each category.

Table 6.4 Classifier performance

Classifiers	**Accuracy Result**
Decision Tree	0.320
K-Nearest Neighbors	0.492
Random Forest	0.502
Multinomial Naïve Bayes	0.515
Logistic Regression	0.552
Support Vector Machine	0.556

According to the results, SVM seems to be the best classifier. It is slightly higher than LR and MNB classifiers. We also evaluated our problem with regression method in

order to find better results. The accuracy results of the best regression methods we obtained, respectively, is kNN(0.352), RF (0.488), Linear Regression (0.587). When we compare the results of regression methods with the classification methods, kNN and RF give worse results even we execute on different test sets. Although Linear Regression gave the best results among all classifiers, we could not go on because we could not find the probability of a class using regression methods we will use in calculations of sentiment analysis below and we could not develop a hybrid method from the results to be obtained.

Now let's look at the performance of each classifier according to four different sentiment analyzes. As we mentioned above, the review-based sentiment analysis method evaluates the entire review to find the category of that review. Because it wants to make sense of the whole review. But sometimes the negative or positive polarization of some words in the reviews is so high that it can affect the whole sentence. In such cases, review-based sentiment analysis categories reviews incorrectly. For this reason, we execute sentence-based sentiment analysis, namely, when classifying a review, we first divide it into sentences, find the probability of each class coming in each sentence, and then take the average of the results. The training set used for review-based is also used for sentence-based. Since the reviews that make up each class are passed through various preprocesses (ngrams, deleting common words, tf/idf), they have specific features. In the sentence based method, each sentence obtained from a review is perceived as a new review. The third one is TextBlob which is a lexicon-based method and it is a Python library for sentiment analysis. It polarizes sentences in the range of [-1, 1]. Namely, -1 means that it is a negative statement and 1 is vice versa. In our study we use TextBlob to predict star ratings of restaurants' reviews in a Likert scale.

The last sentiment analysis method is a hybrid that combines three other methods. For the hybrid sentiment analysis, we apply two different approaches. In the first approach, each sentiment analysis method makes a classification for a review. However, we compare the probability values they calculate when classifying. For instance, suppose we are trying to find out the relevant review belongs to which class. Review-based states that %90 probability is a "5" stars review, sentence-based states that %60 probability is a "4" stars review, and dictionary-based states %40 probability is a "2"

stars review. In this case, we take the highest probability of prediction. We assume that the related sentiment analysis method is more confident in its calculations.

To compare the relevant sentiment analysis methods with each other we select 1000 reviews written about restaurants in total. In order to see how well each method works in each category, we receive 200 reviews from each category.

Table 6.5 The average distance between real ratings and review ratings based on first approach

Classifiers / Sent. A. Methods	RB	SB	DB	HB
Decision Tree	1.097	1.144	**1.060**	1.109
k-Nearest Neighbors	0.741	**0.694**	1.060	0.724
Random Forest	**0.698**	0.974	1.060	0.741
Multinomial Naïve Bayes	0.609	0.610	1.060	**0.586**
Logistic Regression	**0.597**	0.763	1.060	0.601
Support Vector Machine	0.551	0.727	1.060	**0.531**
Average Error	0.716	0.868	1.060	**0,714**

As we mentioned before, the dictionary-based method is a ready application for sentiment analysis. Namely, it is not executed by each classifier. It is placed for display only on the same table. Table 6.5. shows the average distance between the real values of the ratings and the values calculated by the methods based on review texts. That is to say, how close the real rating values are to results based on review texts. According to the results:

-The DB method gives worse results than other methods compared to all other classifiers, except that it gives the best result in the DT classifier.

- The SB method give worse results than RB and HB methods according to all other classifiers, except that it gives the best result in the kNN classifier compared to all other methods.

-When we come to the comparison of RB and HB, we see that the situation changes according to the classifier used. However, we see the best result in Table V. is given by the HB method using the SVM classifier.

- When we look at the average error rate according to the results of all classifiers, we see that the best result is HB, RB, SB and then DB, respectively.

In order to see that the results do not change according to the selected test set, we applied it on different test sets. You can see the results on another test set below.

Table 6.6 Another example for the distance between real ratings and review ratings based on first approach

Classifiers / Sent. A. Methods	RB	SB	DB	HB
Decision Tree	1.130	1.150	**1.105**	1.134
k-Nearest Neighbors	0.708	**0.644**	1.105	0.689
Random Forest	**0.709**	0.936	1.105	0.751
Multinomial Naïve Bayes	0.608	0.607	1.105	**0.582**
Logistic Regression	**0.574**	0.752	1.105	0.589
Support Vector Machine	0.538	0.705	1.105	**0.518**
Average Error	0.711	0.799	1.105	**0,709**

As can be seen from Table 6.6, when we applied our methods on a second test set, and we got similar results.

For the second approach that we use to calculate the star rating of a review by the hybrid sentiment analysis, firstly, as in the first approach, the three other models calculate the class of the review, and then the class of the review is determined by the majority of three models, regardless of the probability ratio of the results. That is, at least two methods need to indicate that the review is in the same class. If each method finds a different class for the text, then the first approach is used to determine the class of the review based on the probability ratio. To make comparison, we used the same data set in Table 6.6. where we got the results using the first approach.

Table 6.7 The distance between real ratings and review ratings based on second approach

Classifiers / Sent. A. Methods	RB	SB	DB	HB
Decision Tree	1.130	1.150	1.105	**1.040**
k-Nearest Neighbors	0.708	**0.644**	1.105	0.663
Random Forest	**0.709**	0.936	1.105	0.714
Multinomial Naïve Bayes	0.608	0.607	1.105	**0.600**
Logistic Regression	**0.574**	0.752	1.105	0.629
Support Vector Machine	0.538	0.705	1.105	**0.530**
Average Error	0.711	0.799	1.105	**0,696**

As can be seen from Table 6.7. when we compare with the first approach, HB sentiment analysis method gives better results with DT, kNN and RF classifiers, while it gives worse results with MNB, LR and SVM classifiers. In order to see that the results do not

change according to the selected test set, we applied it on different test sets multiple times, but we obtained similar results.

As we mention, our goal is to find out how close the real values of the ratings between the values calculated by the methods based on review texts. But what about capturing the real values according to the calculations that each method draws from the review texts? In other words, what is the precision values of each method? Because for some platforms, it may be important how many reviews are correctly classified, rather than how close they are to real values. Let's criticize this situation checking by the SVM classifier which is the best classifier that gives the closest results.

As can be seen from Table 6.8,

-The DB method seems to be much worse than other methods at determining the stars of the reviews. In particular, almost three quarters of 2-, 3- and 4-star reviews are misclassified.

-While the SB method seems to be quite successful in determining 1- and 5-star reviews compared to other methods, this success decreases in 2-, 3- and 4-star reviews.

Table 6.8 Precision values of each method based on Support Vector Machines

Sentiment Analysis Methods	Class 1	Class 2	Class 3	Class 4	Class 5	Total
Review-based	128/200	105/200	100/200	105/200	122/200	560/1000
Sentence-based	177/200	41/200	62/200	71/200	155/200	506/1000
Dictionay-based	96/200	54/200	42/200	55/200	142/200	389/1000
Hybrid-based	138/200	108/200	99/200	109/200	130/200	584/1000

-When we look at the RB method, the number of correct predictions and the almost equal division of this number into each class shows that this method is successful than the DB and SB methods.

-When we look at the HB method, it gives slightly better results than the RB method. It is also another success that the number of correct predictions is distributed almost equally to each class. We get these results using the first hybrid approach. Let's check the second approach of hybrid method that there is any big difference.

Table 6.9 Precision values of hybrid-based method based on second approach using Support Vector Machines

Sentiment Analysis Methods	Class 1	Class 2	Class 3	Class 4	Class 5	Total
Hybrid-based	133/200	107/200	100/200	105/200	127/200	572/1000

As can be seen from Table 6.9, as the previous evaluation, the results of the second method and the equal distribution of the number of correct predictions were similar, but slightly worse than the first method.

6.5 Conclusion

This chapter discussed the effects of both the classifiers and sentiment analysis approaches to predict fine-grained restaurants' review rating stars. For this purpose, we wanted to show which one is the best for the related fields on the five class-based review rating stars and whether there is a better sentiment analysis approach as we called hybrid-based.

The results we have obtained from the researches to date show that the success of each classifier can change in every chosen field. In this study, we observed that the SVM classifier gives better results compared to other classifiers in determining the star ratings of the restaurants based on the text of the reviews. In fact, the point to be considered here is which classifier will perform better according to the sentiment analysis approach chosen. Otherwise, as seen in Table 6.4., we observe that the MNB and LR also performed closely, without selecting the sentiment analysis approach.

When we come to sentiment analysis approaches, we see that the DB method gives worse results compared to other methods, except that it gives the best result in the DT classifier. This shows that TextBlob, a simple API by Python, is not very good at classifying short text in a fine-grained star ratings compared to other methods. The SB method seems good in terms of the number of reviews correctly it classifies, but it is not successful in distributing this number equally to each class. The RB method gives the best results in classifiers such as RF and LR compared to other methods. In addition, the number of correct predictions and the almost equal division of this number into each class shows that this method is successful than the DB and SB methods. Similarly, HB

method gives the best results in MNB and SVM classifiers compared to other methods. With its result in the SVM classifier, it gives the best result among all other classifiers. It also gives the best result in the average error rate of all classifiers. Apart from this, as we can see in Table VIII., it gives the best result in the number of correct predictions and the success of distributing this number almost equally to each class.

We hope that our proposed hybrid-based sentiment analysis approach and the experimental study results will help future researches according to the chosen field.

7. CONCLUSIONS

It is obvious that e-commerce, whose importance has increased more with the Covid-19 pandemic, will push companies to new fields of work in analyzing ever-changing consumer needs and behaviors. In this sense, in order to analyze the users, besides the behaviors of the users in e-commerce environments, many factors such as the society, living conditions, spending habits, ways of spending time, communication forms and channels, and opportunities to access reliable and effective information should be examined. However, since studying human behavior as a whole requires a great deal of effort and time or is impossible in the Internet environment, the problem has been broken down into smaller parts to reach the whole. In other words, the necessary and sufficient information is collected by examining the user's small behaviors such as clicking, watching, listening, buying, sharing with other users. As a result, this information is used to predict the next behavior of the user concerned. We can say that one of the tools that will analyze this information best is the recommendation systems. It has met and will continue to meet the needs of many e-commerce platforms to a large extent with many practical and simple methods. However, due to the diverse e-commerce environments, existing traditional recommendation systems have become unable to meet the needs in this field. In this context, our research aimed to eliminate

the problems in this area to some extent by offering different methods such as trust based, time decay based, review based recommendations.

In Chapter 3, we propose a method based on trustful users to troubleshoot fake accounts especially for e-commerce platforms that offer services such as restaurant, café, hotel, etc. Accordingly, we calculate the trust value of each user by evaluating the relationships between users in the database. Then, while calculating the rating score of each product, we enable the users who rated the product to affect the result according to the trust value. Another suggestion is to calculate based on the experience of the most trusted users in the system, especially for the rating score of the products that are not rated or rated by a few numbers of users. The results we obtained show that the products that are rated by many users and reached their real values are close to our score, and that our method is efficient and can be used in related e-commerce platforms. In addition, it has been observed that challenges such as sparsity and robustness experienced in the recommendation systems can be solved to some extent.

In Chapter 4, a time-based recommendation system has been proposed to prevent the unfair scoring system experienced in many e-commerce platforms, especially providing services such as restaurants, cafes, hotels. It is determined that many companies can not reach the value they deserve due to the ratings given to their workplaces years ago, or they get the value they don't deserve. But we know that the hotter ratting we get for such businesses, the healthier information we get. In order to overcome this situation, a method considering the times of the ratings given by users has been proposed. Accordingly, when calculating the rating score of a product, the result is reached by evaluating the date of the ratings given by each user who rated the product. Thus, the closer the rating given by a user to the present day, the more effective it is to calculate the rating score of the product.

In Chapter 5, to predict fine-grained review rating stars, we focus on the impact of simple text preprocessing decisions especially on restaurant reviews. According to the experimental results, a simple stopwords elimination, lowercasing, removing common words, and lastly the combination of 1-to-3 N-grams perform better than other preprocessing methods for improving the classification accuracy of the five class-based review rating stars. Besides this, results show that the effects of the preprocessing methods can change in any domain. For this reason, all the possible preprocessing

methods should be considered to apply before used in any application. And applying the order of the preprocessing methods can also be important.

In Chapter 6, another method is proposed in which we can ensure that the products get the value they deserve. This method is the sentiment analysis that includes the calculations we made based on the reviews made about the product while calculating the rating score of the product. The proposed hybrid method is aimed to find a better result by using the strengths of existing review-based, sentence-based, and dictionary-based sentiment analysis. Besides, the results of our experimental studies also led to the comparison of the sentiment analysis methods for the quality of the multi-class star rating challenge, specifically on restaurant reviews. According to the results, the average distance between real ratings and review ratings based on the proposed hybrid sentiment analysis method gives the best results using the SVM classifier compared to other methods. it also gives the best result in the average error rate of all classifiers. Apart from this, it gives the best result in the number of correct predictions and the success of distributing this number almost equally to each class.

As a result, we think that all the methods we recommend can play an important role in improving the quality of the recommendation systems. In particular, in order to provide more reliable and effective information to users, it can be presented to the users by calculating multiple product rating scores on the same platform. Thus, the users can make the final decision and prevent the possible frustration of the users. In other words, while giving the general average score of a product, it can also be shown according to the trusted users, time decay of the ratings, and review based. Therefore, users can get healthier information by checking the rating scores of the products based on different methods.

REFERENCES

[1] P. Winters and M. Zeller, "Social Media, Recommendation Engines and Real-Time Model Execution: A Practical Case Study," 2011. [Online]. Available: https://www.knime.org/files/knime_zementis_white_paper.pdf. [Accessed 10 August 2015].

[2] N. Tintarev, "Explaining recommendations," Aberdeen, 2009.

[3] "tf-idf," Wikimedia Foundation, 11 September 2015. [Online]. Available: https://en.wikipedia.org/wiki/Tf%E2%80%93idf. [Accessed 12 December 2016].

[4] The Power of Reviews, "How ratings and reviews influence the buying behavior of the modern consumer," November 2014. [Online]. Available: https://www.powerreviews.com/wp-content/uploads/2016/04/PowerofReviews_2016.pdf. [Accessed 1 June 2020].

[5] A. Ahsan, "Consumer ratings-reviews and its impact on consumer purchasing behavior," KTH Royal Institute of Technology, Stockholm, 2017.

[6] N. I. Holleschovsky and E. Constantinides, "Impact of online product reviews on purchasing decisions," International Conference on Web Information Systems and Technologies, 2016.

[7] S. Deng, L. Huang and G. Xu, "Social network-based service recommandation with trust enhancement," *Expert Systems with Applications*, pp. 8075-8084, 2014.

[8] P. Melville ve V. Sindhwani, «Recommender Systems,» 22 April 2010. [Çevrimiçi]. Available: http://www.prem-melville.com/publications/recommender-systems-eml2010.pdf. [Erişildi: 21 August 2015].

[9] S. Owen, R. Anil, T. Dunning and E. Friedman, Mahout in Action, Shelter Island: Manning Publications Co., 2012.

[10] "Pearson's Correlation Coefficient," University of the West of England, [Online]. Available: http://learntech.uwe.ac.uk/da/Default.aspx?pageid=1442. [Accessed 1

October 2015].

[11] J. D. Ullman, «Clustering,» 2 October 2015. [Çevrimiçi]. Available: http://infolab.stanford.edu/~ullman/mmds/ch7.pdf.

[12] Q. Wang, J. D. Raj and R. LVN, "Recommending News Articles using Cosine Similarity Function," 2014. [Online]. Available: http://support.sas.com/resources/papers/proceedings14/1886-2014.pdf. [Accessed 13 November 2015].

[13] J. B. Schafer, D. Frankowski, J. Herlocker and S. Sen , "Collaborative Filtering Recommender Systems," 2006. [Online]. Available: http://faculty.chas.uni.edu/~schafer/publications/CF_AdaptiveWeb_2006.pdf. [Accessed 20 February 2015].

[14] Y. Chen, C. Wu, M. Xie and X. Guo, "Solving the Sparsity Problem in Recommender Systems Using Association Retrieval," *JOURNAL OF COMPUTERS*, pp. 1896-1902, 9 September 2011.

[15] M. A. Ghazanfar and A. Prugel-Bennett, "A Scalable, Accurate Hybrid Recommender System," 10 January 2010. [Online]. Available: http://eprints.soton.ac.uk/268430/1/Scalable_accurate_HRS.PDF. [Accessed 12 May 2016].

[16] X. Su and T. M. Khoshgoftaar, "A Survey of Collaborative Filtering Techniques," *Advances in Artificial Intelligence*, 3 August 2009.

[17] P. Adamopoulos and A. Tuzhilin, "Probabilistic Neighborhood Selection in Collaborative Filtering Systems," *ACM Conference on Recommender Systems RECSYS 2014*, 6 October 2014.

[18] S. (. K. Lam and J. Riedl, "Shilling Recommender Systems for Fun and Profit," in *WWW2004*, New York, 2004.

[19] M. Işık, H. Dağ ve Y. Işıl, «E-TİCARET SİSTEMLERİ İÇİN BİR ÖNERİ SİSTEMİ: MAHOUT,» %1 içinde *YBS.2014*, İstanbul, 2014.

[20] M. Trujillo, M. Millan and E. Ortiz, "A Recommender System Based on Multi-features," in *ICCSA 2007*, Berlin, 2007.

[21] H. Ma, H. Yang, M. R. Lyu and I. King, "SoRec: social recommendation using probabilistic matrix factorization," in *CIKM'08*, California, 2008.

[22] D. Shin, J.-w. Lee, J. Yeon and S.-g. Lee, "Context-Aware Recommendation by Aggregating User Context," in *IEEE Conference on Commerce and Enterprise Computing*, Vienna, 2009.

[23] M. Jamali and M. Ester, "Modeling and comparing the influence of neighbors on the behavior of users in social and similarity networks," in *IEEE International Conference on Data Mining Workshops*, Sydney, 2010.

[24] N. Zheng ve Q. Li, «A recommender system based on tag and time information for social tagging systems,» 2010. [Çevrimiçi]. Available: http://www.sciencedirect.com/science/article/pii/S0957417410010882.

[25] S. J. Yu, "The dynamic competitive recommendation algorithm in social network services," *Information Sciences*, pp. 1-14, 6 November 2011.

[26] H.-N. Kim, A. Alkhaldi, A. El Saddik and G.-S. Jo, "Collaborative user modeling with user-generated tags for social recommender systems," *Expert*

Systems with Applications, pp. 8488-8496, 2011.

[27] J. Zhang, Y. Wang and J. Vassileva, "SocConnect: A personalized social network aggregator and recommender," *Information Processing and Management,* pp. 721-737, 6 Ekim 2012.

[28] F. Ullah, G. Sarwar and S. Lee, "Social network and device aware personalized content recommendation," *Conference on Electronics, Telecommunications and Computers– CETC 2013,* p. 528 – 533, 2014.

[29] Z. Sun, L. Han, W. Huang, X. Wang, X. Zeng, M. Wang and H. Yan, "Recommender systems based on social networks," *The Journal of Systems and Software,* pp. 109-119, 5 Ekim 2014.

[30] L. Yu-sheng, S. Mei-na, E. Hai-hong and S. Jun-d, "Social recommendation algorithm fusing user interest social network," July 2014. [Online]. Available: https://www.researchgate.net/publication/265128498. [Accessed 11 May 2016].

[31] X. Han, L. Wang, N. Crespi, S. Park and Á. Cuevas, "Alike people, alike interests? Inferring interest similarity in online social networks," *Decision Support Systems,* pp. 92-106, 9 Aralık 2014.

[32] T. Yuan, J. Cheng, X. Zhang, Q. Liu and H. Lu, "How friends affect user behaviors? An exploration of social relation analysis for recommendation," *Knowledge-Based Systems,* pp. 70-84, 12 August 2015.

[33] A. J. Chaney, D. M. Blei and T. Eliassi-Rad, "A Probabilistic Model for Using Social Networks in Personalized Item Recommendation," in *RecSys '15: Proceedings of the 9th ACM Conference on Recommender Systems*, Vienna, 2015.

[34] M. Gan, "COUSIN: A network-based regression model for personalized recommendations," *Decision Support Systems,* pp. 58-68, 11 December 2015.

[35] F. Colace, M. D. Santo, L. Greco, V. Moscato and A. Picariello, "A collaborative user-centered framework for recommending items in Online Social Networks," *Computers in Human Behavior,* pp. 694-704, 5 Ocak 2015.

[36] A. H. Celdrán, M. G. Pérez, F. J. G. Clemente and G. M. Pérez, "Design of a recommender system based on users' behavior and collaborative location and tracking," *Journal of Computational Science,* pp. 83-94, 10 December 2015.

[37] C. Yang, Y. Zhou and D. M. Chiu, "Who are like-minded: mining user interest similarity in online social networks," *Proceeding of the AAAI Conference on Web and Social Media,* pp. 731-734, 7 March 2016.

[38] C. Biancalana, F. Gasparetti, A. Micarelli, A. Miola and G. Sansonetti, "Context-aware movie recommendation based on signal processing and machine learning," *Proceedings of the 2nd Challenge on Context-Aware Movie Recommendation,* pp. 5-10, October 2011.

[39] D. Fijałkowski and R. Zatoka, "An architecture of a Web recommender system using social network user profiles for e-commerce," in *Proceedings of the Federated Conference on Computer Science and Information Systems*, Wrocław, 2011.

[40] W. Carrer-Neto, M. L. Hernández-Alcaraz, R. Valencia-García and F. García-Sánchez, "Social knowledge-based recommender system. Application to the movies domain," *Expert Systems with Applications, vol. 39, no. 12,* pp. 10990-

11000, September 2012.

[41] P. Bedi, H. Kaur and S. Marwaha, "Trust based recommender system for the semantic web," *Proceedings of the 20th international joint conference on Artifical intelligence*, pp. 2677-2682, January 2007.

[42] J. O'Donovan and B. Smyth, "Trust in recommender systems," *Proceedings of the 10th international conference on Intelligent user interfaces*, pp. 167-174, January 2005.

[43] M. Jamali and M. Ester, "TrustWalker: A Random walk model for combining trust-based and item-based recommendation," *Proceedings of the 15th ACM SIGKDD international conference*, p. 2009, June 28- July 1 2009.

[44] H. Ma, I. King and M. R. Lyu, "Learning to recommend with social trust ensemble," *Proceedings of the 32nd international ACM SIGIR conference*, pp. 203-210, 19-23 July 2009.

[45] N. Lathia, S. Hailes and L. Capra, "Trust-based collaborative filtering," *International Conference on Trust Management*, pp. 119-134, 2008.

[46] C.-W. Hang and M. P. Singh, "Trust-based recommendation based on graph similarity," in *Proceedings of the 13th International Workshop on Trust in Agent Societies (TRUST)*, Toronto, Canada, 2010.

[47] Y.-M. Li, C.-T. Wu and C.-Y. Lai, "A social recommender mechanism for e-commerce: Combining similarity, trust, and relationship," *Decision Support Systems*, pp. 740-752, 13 Mart 2013.

[48] C. Chen, J. Zeng, X. Zheng and D. Chen, "Recommender system based on social trust relationships," *2013 IEEE 10th International Conference on e-Business Engineering*, September 2013.

[49] B. Yang, Y. Lei, D. Liu and J. Liu, "Social collaborative filtering by trust," *IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE*, pp. 1633-1647, August 2017.

[50] D. O'Doherty, S. Jouili and P. V. Roy, "Trust-Based Recommendation: an Empirical Analysis," in *Sixth ACM Workshop on Social Network Mining and Analysis (SNA-KDD 2012)*, 2012.

[51] M. G. Ozsoy and F. Polat, "Trust Based Recommendation Systems," in *Proceedings of the 2013 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, Niagara Ontario Canada, 2013.

[52] H. Zhong, S. Zhang, Y. Wang and Y. Shu, "Study on Directed Trust Graph Based Recommendation for E-commerce System," *INTERNATIONAL JOURNAL OF COMPUTERS COMMUNICATIONS & CONTROL*, pp. 510-523, August 2014.

[53] R. Chamsi Abu Quba, S. Hassas, U. Fayyad, M. Alshomary and C. Gertosio, "iSoNTRE: the Social Network Transformer into Recommendation Engine," in *2014 IEEE/ACS 11th International Conference on Computer Systems & Applications (AICCSA)*, Doha, Qatar, 2014.

[54] G. Guo, J. Zhang, D. Thalmann, A. Basu and N. Yorke-Smith, "From Ratings to Trust: an Empirical Study of Implicit Trust in Recommender Systems," in *SAC '14: Proceedings of the 29th Annual ACM Symposium on Applied Computing*, Gyeongju Republic of Korea, 2014.

[55] D. H. Alahmadi and X.-J. Zeng, "ISTS: Implicit social trust and sentiment based approach to recommender systems," *Expert SystemsWithApplications*, pp. 8840-8849, 2015.

[56] S. Deng, L. Huang, Y. Yin and W. Tang, "Trust-based service recommendation in social network," *Applied Mathematics & Information Sciences*, pp. 1567-1574, 1 May 2015.

[57] F. Keikha, M. Fathian and M. R. Gholamian, "TB-CA: A hybrid method based on trust and context-aware for recommender system in social networks," *Management Science Letters*, p. 471–480, 18 June 2015.

[58] H. Wu, K. Yue, Y. Pei, B. Li, Y. Zhao and F. Dong, "Collaborative topic regression with social trust ensemble for recommendation in social media systems," *Knowledge-BasedSystems*, p. 1–12, 16 January 2016.

[59] N. ÇETİN and N. ORHUN, Apache Mahout Scalable Machine Learning and Data Mining, Eskişehir: Anadolu University, 1998.

[60] R. Tanase and R. Radu , "Linear Algebra," 2009. [Online]. Available: http://www.math.cornell.edu/~mec/Winter2009/RalucaRemus/Lecture1/lecture1.html. [Accessed 28 October 2016].

[61] M. IŞIK, Pagerank && Trustrank, İstanbul: İkinci adam yayınları, 2013.

[62] S. Brin, L. Page, R. Motwami and T. Winogard, "The PageRank citation ranking: Bringing order to the web," Stanford University, Computer Science Department, 1999.

[63] R. S. Wills, *When Rank Trumps Precision: Using The Power Method to Compute Google's PageRank*, Raleigh: Nort Carolina State University, Dept. of Mathematics, 2007.

[64] "Trust marks report 2013," Linnea Persson, European Consumer Centre Sweden, 2013.

[65] T. Nwaogu, V. Simittchieva, M. Whittle and M. Richardson, "Study on Online Consumer Reviews in the Hotel Sector," European Commission Directorate General for Health and Consumers (DG SANCO), 2014.

[66] Q. T. Lee, Y. Park and Y.-T. Park, "A time-based approach to effective recommender systems using implicit feedback," *Expert Systems with Applications*, pp. 3055-3062, 2008.

[67] R. Raju, I. Pradeep, I.Bhagyasri, P. Praneetha and P. Teja, "Recommender systems for e-commerce: novel parameters and issues," *International Journal of Advanced Research in Computer Science and Software Engineering*, pp. 248-253, September 2013.

[68] Y. Zhang, M. Zhang, Y. Zhang, G. Lai, Y. Liu, H. Zhang and S. Ma, "Daily-aware personalized recommendation based on feature-level time series analysis," in *WWW '15: Proceedings of the 24th International Conference on World Wide Web*, Florence, Italy, 2015.

[69] X. Jiang and Y. Zhang, "Dynamic item-based recommendation algorithm with time decay," *2010 Sixth International Conference on Natural Computation ,* pp. 241-247, 2010.

[70] "The 2018 Amazon Shopper Behaivor Study," 2018. [Online]. Available: http://learn.cpcstrategy.com/rs/006-GWW-889/images/2018-Amazon-Shopper-

Behavior-Study.pdf. [Accessed 10 September 2018].

[71] "2018 ReviewTrackers Online Reviews Stats and Survey," Review Trackers, [Online]. Available: https://www.reviewtrackers.com/reports/online-reviews-survey/. [Accessed 19 September 2018].

[72] "TripBarometer 2016," June 21 – July 8 2016. [Online]. Available: https://www.tripadvisor.com/TripAdvisorInsights/wp-content/uploads/2018/01/TripBarometer-2016-Traveler-Trends-Motivations-Global-Findings.pdf. [Accessed 15 August 2018].

[73] P. Sharma, A. Agrawal, L. Alai and A. Garg, "Challenges and Techniques in Preprocessing for Twitter Data," *International Journal of Engineering Science and Computing,*, pp. 6611-6613, April 2017.

[74] K. V. Ghag and K. Shah, "Comparative Analysis of Effect of Stopwords Removal on Sentiment Classification," *IEEE International Conference on Computer, Communication and Control*, 2015.

[75] Z. Jianqiang and G. Xiaolin, "Comparison Research on Text Pre-processing Methods on Twitter Sentiment Analysis," *IEEE Access*, pp. 2870-2879, 22 February 2017.

[76] V. Srividhya and R. Anitha, "Evaluating Preprocessing Techniques in Text Categorization," *International Journal of Computer Science and Application Issue*, pp. 49-51, 2010.

[77] J. Camacho-Collados and M. T. Pilehvar, "On the Role of Text Preprocessing in Neural Network Architectures:," *Cornell University Library*, 23 August 2018.

[78] K. V. Ghag and K. Shah, "Optimising Sentiment Classification using Preprocessing Techniques," *International Journal of IT & Knowledge Management*, pp. 61-70, Jan-Jun 2015.

[79] R. Gull, U. Shoaiba, S. Rasheed, W. Abid and B. Zahoor, "Pre Processing of Twitter's Data for Opinion Mining in Political Context," *20th International Conference on Knowledge Based and Intelligent Information and Engineering*, pp. York, United Kingdom, 5-7 September 2016.

[80] Z. Jianqiang, "Pre-processing Boosting Twitter Sentiment Analysis?," *IEEE International Conference on Smart City/SocialCom/SustainCom*, pp. 748-753, 19-21 December 2015.

[81] I. Safeek and M. R. Kalideen, "Preprocessing on Facebook Data for Sentiment Analysis," in *7th International Symposium 2017 on "Multidisciplinary Research for Sustainable Development"*, Oluvil, Sri Lanka, 2017.

[82] S. Vijayarani, J. Ilamathi and Nithya, "Preprocessing Techniques for Text Mining - An Overview," *International Journal of Computer Science & Communication Networks*, pp. 7-16, 2015.

[83] I. Hemalatha, G. P. S. Varma and A. Govardhan, "Preprocessing the Informal Text for efficient Sentiment Analysis," *International Journal of Emerging Trends & Technology in Computer Science (IJETTCS)*, pp. 58-61, July-August 2012.

[84] N. P.Katariya and M. S. Chaudhari, "Text Preprocessing For Text Mining Using Side Information," *International Journal of Computer Science and Mobile Applications*, pp. 01-05, January 2015.

[85] T. Singh and M. Kumari, "The Role of Text Pre-processing in Sentiment Analysis," in *Twelfth International Multi-Conference on Information Processing-2016 (IMCIP-2016)*, 2016.

[86] A. S. Nayak, A. P. Kanive, N. Chandavekar and B. R, "Survey on Pre-Processing Techniques for Text Mining," *International Journal Of Engineering And Computer Science*, pp. 16875-16879, 6 June 2016.

[87] A. Krouska, C. Troussas and M. Virvou, "The effect of preprocessing techniques on Twitter Sentiment Analysis," in *2016 7th International Conference on Information, Intelligence, Systems & Applications (IISA)*, Chalkidiki, Greece, 2016.

[88] H. M. Zin, N. Mustapha, M. A. A. Murad and N. M. Sharef, "The Effects of Pre-Processing Strategies in Sentiment Analysis of Online Movie Reviews," in *The 2nd International Conference on Applied Science and Technology 2017 (ICAST'17)*, 2017.

[89] J. Pomikalek and R. Rehurek, "The Influence of preprocessing parameters on text categorization," *World Academy of Science, Engineering and Technology*, no. 33, January 2007.

[90] A. Schofield, M. Magnusson, L. Thompson and D. Mimno, "Understanding Text Pre-Processing for Latent Dirichlet Allocation," *Widening Natural Language Processing* , 2017.

[91] M. Fan and M. Khademi, "Predicting a Business' Star in Yelp from Its Reviews' Text Alone," January 2014. [Online]. Available: https://www.researchgate.net/publication/259578317_Predicting_a_Business_Star_in_Yelp_from_Its_Reviews_Text_Alone. [Accessed 12 June 2018].

[92] R. Duwairi and M. El-Orfali, "A study of the effects of preprocessing strategies on sentiment analysis for Arabic text," *Journal of Information Science*, pp. 501-513, 12 May 2014.

[93] M. K. Saad, *The Impact of Text Preprocessing and Term Weighting on Arabic Text Classification*, Computer Engineering Department - The Islamic University - Gaza, 2010.

[94] A. K. Uysal and S. Gunal, "The impact of preprocessing on text classification," *Information Processing and Management*, pp. 104-112, 16 September 2013.

[95] M. O. Shiha and S. Ayvaz, "The Effects of Emoji in Sentiment Analysis," *International Journal of Computer Electrical Engineering*, pp. 360-369, June 2017.

[96] K. Wegrzyn-Wolska, L. Bougueroua, . H. Yu and J. Zhong, "Explore the Effects of Emoticons on Twitter Sentiment Analysis," *Computer Science and Information Technology*, pp. 65-77, 2016.

[97] Y. Park and R. J. Byrd, "Hybrid text mining for finding abbreviations and their definitions," IBM Thomas J. Watson Research Center.

[98] A. Kaur, P. Singh and S. Rani, "Spell Checking and Error Correcting System for text paragraphs written in Punjabi Language using Hybrid approach," *International Journal Of Engineering And Computer Science*, pp. 8030-8032, September 2014.

[99] N. Bertoldi, M. Cettolo and M. Federico, "Statistical Machine Translation of

Texts with Misspelled Words," *Association for Computational Linguistics,* pp. 412-419, June 2010.

[100] P. O. Müller, I. Ohnheiser, S. Olsen and F. Reiner, "Multi-word Expressions," October 2011. [Online]. Available: http://neon.niederlandistik.fu-berlin.de/static/mh/Huening_Schluecker_2015_ms.pdf. [Accessed 11 August 2018].

[101] M. Constant, A. Sigogne and P. Watrin, "Discriminative Strategies to Integrate Multiword Expression Recognition and Parsing," *Association for Computational Linguistics,* pp. 204-212, July 2012.

[102] M. Schonlau, N. Guenther and I. Sucholutsky, "Text mining with ngram variables," *The Stata Journal,* pp. 866-881, 2017.

[103] "European Ecommerce Report 2018 Edition," Ecommerce Operations, Netherlands, 2018.

[104] Y. Wan and M. Nakayama, "The Reliability of Online Review Helpfulness," *Journal of Electronic Commerce Research,* pp. 179-189, 2014.

[105] G. Lackermair, D. Kailer and K. Kanmaz, "Importance of Online Product Reviews from a Consumer's Perspective," *Advances in Economics and Business,* pp. 1-5, 2013.

[106] P. JAIN, K. JAIN and P. K. JAIN, "Electronic-commerce and its Global Impact," *Journal of Engineering & Technology,* pp. 1-6, 26 May 2016.

[107] M. Govindarajan, "Sentiment Analysis of Restaurant Reviews Using Hybrid Classification Method," *International Journal of Soft Computing and Artificial Intelligence,* pp. 17-23, May 2014.

[108] Y. Guo and Z. Wang, "Predicting Restaurants' Rating and Popularity Based on Yelp Dataset," 2017. [Online]. Available: http://cs229.stanford.edu/proj2017/final-reports/5244334.pdf. [Accessed 23 12 2018].

[109] M. Yu, M. Xue and W. Ouyang, "Restaurants Review Star Prediction for Yelp Dataset," 2015. [Online]. Available: https://pdfs.semanticscholar.org/43c5/6af59062ac41031ef80b5853cc323a20be26.pdf. [Accessed 22 11 2018].

[110] N. Asghar, "Yelp Dataset Challenge: Review Rating Prediction," 17 May 2016. [Online]. Available: https://arxiv.org/pdf/1605.05362.pdf. [Accessed 12 10 2018].

[111] B. Kapukaranov and P. Nakov, "Fine-Grained Sentiment Analysis for Movie Reviews in Bulgarian," *International Conference Recent Advances in Natural Language Processing,* pp. 266-274, 2015.

[112] A. Ghazvinian, "Star Quality: Sentiment Categorization of Restaurant Reviews," 2007. [Online]. Available: https://pdfs.semanticscholar.org/8afb/157ef26251b42bdf44e5d440e1aaa3297c90.pdf. [Accessed 15 09 2018].

[113] J. L. Lee, P. B. Awayan and E. Mendoza, "A Comparative Study: Different Automatic Approaches of Stars Generation for Reviews," *International Conference on Software and e-Business,* pp. 28-32, 28-30 December 2017.

[114] T. Doan and J. Kalita, "Sentiment Analysis of Restaurant Reviews on Yelp with

Incremental Learning," *IEEE International Conference on Machine Learning and Applications (ICMLA),* 18-20 December 2016.

[115] H. Zhang, X. Liu and K. Ying, "Reviews Usefullness Prediction for Yelp Dataset," 2017. [Online]. Available: https://cseweb.ucsd.edu/classes/wi17/cse258-a/reports/a040.pdf. [Accessed 12 12 2018].

[116] W. Huang and Y. Yu, "Is it truly a 5-Star Movie? Restoring the Movie's Truthful Rating," *IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM),* pp. 1337-1338, 18-21 August 2016.

[117] M. Ochi, M. Okabe and R. Onai, "Rating Prediction using Feature Words Extracted from Customer Reviews," *International ACM SIGIR conference on Research and development in Information Retrieval,* pp. 1205-1206, 24-28 July 2011.

[118] Z. Jin, Q. Li, D. D. Zeng, Y. Zhan, R. Liu, L. Wang and H. Ma, "Jointly Modeling Review Content and Aspect Ratings for Review Rating Prediction," *International ACM SIGIR conference,* 17-21 July 2016.

[119] H. Wang, Y. Lu and C. Zhai, "Latent Aspect Rating Analysis without Aspect Keyword Supervision," *International conference on Knowledge discovery and data mining,* pp. 618-626, 21-24 August 2011.

[120] Y. Xu, W. Lam and R. Fan, "Hidden Aspect Rating Discovery from Text Reviews of E-commerce Web Sites," *Big Data Science International Conference,* 04-07 August 2014.

[121] Y. Xu, X. Wu and Q. Wang, "Sentiment Analysis of Yelp's Ratings Based on Text Reviews," 2009. [Online]. Available: http://cs229.stanford.edu/proj2014/Yun%20Xu,%20Xinhui%20Wu,%20Qinxia%20Wang,%20Sentiment%20Analysis%20of%20Yelp's%20Ratings%20Based%20on%20Text%20Reviews.pdf. [Accessed 12 12 2018].

[122] H. Chen, P. Wu, N. Yi, S. Li and X. Huang, "Fine-grained Sentiment Analysis of Chinese Reviews Using LSTM Network," *Journal of Engineering Science and Technology Review,* pp. 174-179, 25 February 2018.

[123] S. K. Chauhan, A. Goel, P. Goel, A. Chauhan and M. K. Gurve, "Research on Product Review Analysis and Spam Review Detection," *International Conference on Signal Processing and Integrated Networks (SPIN),* 2-3 February 2017.

[124] U. Kumari, A. Sharma and D. Soni, "Sentiment Analysis of Smart Phone Product Review using SVM Classification Technique," *International Conference on Energy, Communication, Data Analytics and Soft Computing (ICECDS),* 1-2 August 2017.

[125] R. R. L. Barbosa , S. Sánchez-Alonso and M. A. Sicilia-Urban , "Evaluating Hotels Rating Prediction Based on Sentiment Analysis Services," *Aslib Journal of Information Management,* pp. 392-407, January 2015.

[126] H.-L. Yang and A. F. Chao, "Sentiment Annotations for Reviews: An Information Quality Perspective," *Online Information Review,* pp. 579-594, 09 April 2018.

[127] G. Vinodhini and R. Chandrasekaran, "A comparative Performance Evaluation

of Neural Network Based Approach for Sentiment Classification of Online Reviews," *Journal of King Saud University - Computer and Information Sciences,* pp. 2-12, January 2016.

[128] X. Fang and J. Zhan, "Sentiment Analysis Using Product Review Data," *Journal of Big Data,* 16 June 2015.

[129] M. Malik, S. Habib and P. Agarwal, "A Novel Approach to Web-Based Review Analysis Using Opinion Mining," *Procedia Computer Science,* pp. 1202-1209, 2018.

[130] S. Rudolph, "The Impact of Online Reviews on Customers' Buying Decisions," 25 July 2015. [Online]. Available: https://www.business2community.com/infographics/impact-online-reviews-customers-buying-decisions-infographic-01280945#xJ5oI7soj1AAJYlv.97. [Accessed 10 01 2019].

[131] "Local Consumer Review Survey," BrightLocal, 2018. [Online]. Available: https://www.brightlocal.com/learn/local-consumer-review-survey/#local-business-review-habits. [Accessed 22 01 2019].

[132] A. Hogenboom, D. Bal, F. Frasincar, M. Bal, F. d. Jong and U. Kaymak, "Exploiting Emoticons in Sentiment Analysis," *ACM Symposium on Applied Computing,* 18-22 March 2013.

[133] P. Chakriswaran, D. R. Vincent, . K. Srinivasan, V. Sharma, C.-Y. Chang and D. G. Reina, "Emotion AI-Driven Sentiment Analysis: A Survey, Future Research Directions, and Open Issues," *Applied Sciences,* 12 December 2019.

[134] Y.-J. Tai and H.-Y. Kao, "Automatic Domain-Specific Sentiment Lexicon Generation with Label Propagation," *Information Integration and Web-based Applications & Services,* 2-4 December 2013.

[135] M. IŞIK and H. DAĞ, "The impact of text preprocessing on the prediction of review ratings," *Turkish Journal of Electrical Engineering and Computer Sciences,* p. 1405 – 1421, 08 May 2020.

[136] G. Maria Di Nunzio and F. Vezzani, "A Linguistic Failure Analysis of Classification of Medical Publications: A Study on Stemming vs Lemmatization," *CLiC-it,* 2018.

[137] R. Ahuja, A. Chug, S. Kohli, S. Gupta and P. Ahuja, "The Impact of Features Extraction on the Sentiment Analysis," *International Conference on Pervasive Computing Advances and Applications,* p. 341–348, 2019.

[138] A. Tripathy, A. Agrawal and S. K. Rath, "Classification of sentiment reviews using n-gram machine learning approach," *Expert Systems With Applications,* pp. 117-126, 24 March 2016.

[139] L. Wright, "Classifying textual fast food restaurant reviews quantitatively using text mining and supervised machine learning algorithms," East Tennessee State University, 2018.

[140] G. DİLKİ and Ö. DENİZ BAŞAR, "İŞLETMELERİN İFLAS TAHMİNİNDE K- EN YAKIN KOMŞU ALGORİTMASI ÜZERİNDEN UZAKLIK ÖLÇÜTLERİNİN KARŞILAŞTIRILMASI," *Istanbul Commerce University Journal of Science,* pp. 224-233, 2020.

[141] Ö. Akar and O. Güngör, "Classification of multispectral images using Random

Forest algorithm," *Journal of Geodesy and Geoinformation,* pp. 105 - 112, November 2012.

[142] S. Mohod, C. Dhote and V. Thakare, "Modified Approach of Multinomial Naïve Bayes for Text Document Classification," *International Journal of Computer Science & Communication,* pp. 196-200, April - Sep 2015.

[143] A. Ng, "www.joparga3.github.io," March 2018. [Online]. Available: https://joparga3.github.io/standford_logistic_regression/#what-is-logistic-regression. [Accessed 20 February 2019].

[144] R. Baly and H. M. Hajj, "Wafer Classification Using Support Vector Machines," *IEEE Transactions on Semiconductor Manufacturing,* pp. 372-383, August 2012.

[145] M. Awad and R. Khanna, Efficient Learning Machines, Elsevier BV on behalf of Faculty of Engineering, Ain Shams University, 2015, pp. 39-66.

CURRICULUM VITAE

Personal Information
Name and surname: Muhittin IŞIK

Academic Background
Bachelor's Degree Education…: Anadolu University, Eskişehir (Turkey)
Education of Computer and Instructional Technologies
Post Graduate Education……...: Kadir Has University, İstanbul (Turkey)
Information Technology
Foreign Languages…….: English (C2)

Work Experience
Institutions Served and Their Dates:
01/09/2017 – 01/07/201
Information Technologies and Software Applications Teacher
İhsan Şerif Elementary School, İstanbul (Turkey)
01/07/2014 – Present
Information Technologies and Software Applications Teacher
Hasköy Secondary School, İstanbul (Turkey)

Contact
Phone:
E-mail Address: