**RECOMMENDATION SYSTEM BASED ON CUSTOMER REVIEWS**

**ABSTRACT**

In recent years, a variety of review-based recommender systems have been developed,with the goal of incorporating the valuable information in user-generated textual reviews into the user modeling and recommending process.

**1.INTRODUCTION**

Recommender systems (RS) have attracted attention in both academia and industry. Such systems help to manage information overload by autonomously gathering information and proactively tailoring it to individual interests (Adomavicius and Tuzhilin 2005), e.g., what product to buy (Amazon), what song to listen to (Last.fm), which hotel to stay in (TripAdvisor ), and so on. Currently, most of the various types of recommender techniques use user-provided ratings to infer user preferences. There are two common memory-based collaborative filtering (CF)approaches (Adomavicius and Tuzhilin 2005; Herlocker et al 2004; Sarwar et al 2001; Schaferet al 2007; Su and hoshgoftaar 2009): the user-based method uses ratings to associate a user with a group of like-minded users and then recommends to the target user a set of items that are enjoyed by her/his neighbors; and the item-based method aims to find items that are similar to those that a user has viewed/purchased before. In contrast, model-based CF systems focus on learning the latent factors that represent users’ inherent preferences over an item’s multiple dimensions (Koren and Bell 2011; Koren et al 2009). Collaborative filtering techniques perform well when there is sufficient rating information (Su and Khoshgoftaar 2009). However, their effectiveness is limited when the well-known rating sparsity problem occurs, due to the poor coverage of recommendation space (Garcia Esparza et al 2010), or the difficulty in letting users express their preferences as scalar ratings on items (Leung et al 2006). To address this problem, content-based recommender approaches have been developed that rely instead on the content representations of items to locate items that have similar content to items the target user liked (Lops et al 2011; Pazzani and Billsus 2007). Some studies have used other types of user-generated information, such as tags (freely chosen/written keywords) (Marinho et al 2011; Zhao et al 2008), and social relationships (like friendship, membership, and trust relationship) (Beilin and Yi 2013; Chen et al 2013; Yang et al 2012), to augment the accuracy of recommendation. However, these methods are still inadequate, especially when the target user has little historical data. They are also of limited usefulness when the overall data sparsity level is high.Therefore, in this paper, we particularly emphasize user reviews, and provide a comprehensive survey of recent attempts to use the valuable information in reviews to solve the rating sparsity issue. The growing popularity of social and e-commerce media sites has encouraged users tonaturally write reviews describing their assessment of items. These reviews are usually in the form of textual comments that explain why they like or dislike an item based on their usage experiences. The system can capture the multi-faceted nature of a user’s opinions from her/his reviews and hence build a fine-grained preference model for the user, which however cannot be obtained from overall ratings. Empirical findings from marketing and consumer behavior studies have also documented the positive influence of product reviews on the decision processes of newusers (Chatterjee 2001; Chevalier and Mayzlin 2006; Kim and Srivastava 2007).There are increasing efforts to incorporate the rich information embedded in reviews into the process of user modeling and recommendation generation. In particular, information obtained from reviews is likely to benefit recommender systems in the following three ways (Chen and Wang 2013; Garcia Esparza et al 2011; Hariri et al 2011; Jamroonsilp and Prompoon 2013; Leviet al 2012; McAuley and Leskovec 2013; Pero and Horv´ath 2013; Wang et al 2012; Yates et al

2008; Zhang et al 2013).the first type,Section 4.3 summarizes research that has used reviews to enhance ratings so that a preference model can be constructed for a user with few ratings by aligning the review information (such as review topics or feature opinions) with numerical ratings. For the second type, a user’s current preference is often elicited on site when s/he is using the system, so the main focus has been on using review elements to either assist the user to complete the preference (see Section 4.4.3) or enrich the product profile (see Section 5). As an example from product profiles, the comparative opinions extracted from reviews can be helpful for constructing product-to-product comparison relationships and enhancing the ranking quality.Third, when the dataset is not sparse (i.e., in a relatively dense data condition), the reviews can still be useful. They have been used to determine rating quality (with the degree of the review’s helpfulness) (see Section 4.3.1), to help derive users’ context-dependent preferences (with the contextual information extracted from reviews) (see Section 4.3.4), and to learn users’ latent preference factors by considering the aspect opinions mentioned in reviews (see Section 4.4.1).

1. **LITERATURE REVIEW**

**First** **topics**:They can help to deal with the problem of large data sparsity by providing additional information about user preferences. In the extreme case of no ratings being available, the reviews can be used to infer the ratings that CF systems require (see Section 4.2).Second, they can help to solve the cold-start problem for new users. Usually, there are two types of new users: a user with limited experience with the items, who therefore has not provided many ratings; and a user who is totally new to the system. For Frequent terms: Because a review is written in natural language, the most obvious way of analyzing it is to identify frequently used terms. A weighting measure such as TF-IDF (as mentioned in Section 2.1) can be applied to determine how representative each term is in the review. The extracted terms can then be used to characterize the reviewer with a termbased user profile. In (Garcia Esparza et al 2010, 2011), the built profile is leveraged into the content-based approach to generate recommendations (see Section 4.1).

**2. Review topics:** Topics are the aspects of an item that a writer discusses in a review. For example, in Figure 2 (a real hotel review from TripAdvisor 1

), the mentioned topics include the hotel room’s quality, food, gym facility, location. There are two approaches to identifying topics in reviews. The first is the frequency-based approach, which first extracts frequently occurring nouns based on a set of seed words, and then groups the nouns into topics manually or according to a pre-defined dictionary (Musat et al 2013). The second approach is to use a topic modeling technique such as Latent Dirichlet Allocation (LDA) (Blei et al 2003), to automatically uncover hidden topics in review documents. The objective of LDA is to cluster words that co-occur in documents to form topics, so that each document d can be represented

as a K-dimensional topic distribution θd, and each topic k is assigned a word distribution φk to indicate the probability that a particular word is related to it. The discovered review topics can then be used to enhance real ratings in CF based recommending approaches (McAuley and Leskovec 2013; Seroussi et al 2011) (see Section 4.3.2).

**3. Overall opinions:** A user’s sentiment orientation (i.e., positive or negative) towards an item can be inferred from review to represent her/his overall opinion. For instance, for the review in Figure 2, we can infer that the reviewer has an overall positive opinion about this hotel. A simple way to estimate the overall opinion is to aggregate the sentiments of all of the opinion words that are contained in the review (Leung et al 2006; Zhang et al 2013). Alternatively, a machine learning algorithm (such as the naive Bayesian classifier or Support Vector Machine (SVM)) can be adopted to learn the opinion and classify it into a proper sentiment category (Pang et al 2002; Poirier et al 2010b). The inferred overall opinions can then be converted into virtual ratings, which may take the role of real ratings in CF (Poirier et al 2010b; Zhang et al 2013) (see Section 4.2), or be used to enhance real ratings (Pero and Horv´ath 2013) (see Section 4.3.3).

**4. Feature opinions:** In addition to the overall opinion, fine-grained opinions about specific features of an item can also be extracted from reviews. For example, the review sentence (see Figure 2) “Rooms are spacious and luxuriously appointed” expresses the author’s positive sentiment towards the feature “room”. In a raw review, the feature is normally expressed

as a noun or noun phrase, which may refer to a distinct object such as the item itself (e.g., “hotel”), one of its components (e.g., “bedroom” or “bathroom”), its function (e.g.,“service”), or a property of the component (or function) (e.g., “size”). Multiple features can further be mapped to an aspect to indicate an upper-level abstraction (Hu and Liu 2004b; Popescu and Etzioni 2005). For example, the hotel’s features “room,” “size,” and “cleanness” can be projected onto the aspect “room quality”. The typical approaches to feature extraction include statistics based methods, such as one that captures frequently occurring nouns/phrases as feature candidates through association rule mining (Hu and Liu 2004b), a LDA or SVM based method for identifying aspects directly, or a machine learning method based on a lexicalized Hidden Markov Model (L-HMMs) (Jin et al 2009)or Conditional Random Fields (CRFs) (Miao et al 2010; Qi and Chen 2010). The opinionsassociated with features (or aspects) are then identified by looking for nearby adjectives, or through opinion pattern mining (Hu and Liu 2004a; Moghaddam and Ester 2010).

In Section 4.4.1, we show how the feature opinions can be modeled as latent preference factors of users and used to augment model-based CF (Jakob et al 2009; Wang et al 2012); or exploited to derive users’ weight preferences (Chen and Wang 2013; Liu et al 2013) or attribute value preferences (Wang et al 2013), for use in preference-based product ranking (see Sections 4.4.2 and 4.4.3). In addition, they are helpful for building product profiles to increase the ranking quality (Aciar et al 2007; Dong et al 2013b; Yates et al 2008) (see Section 5.1).

**5. Contextual opinions:** The review sentence “first visit to company’s Hong Kong offices” (see Figure 2) provides the contextual information related to this review. As another example,“This camera’s image quality is not good when I used it to take pictures at night,” “at night”is the context, “image quality” is the feature, and “not good” is the opinion that is negative.This kind of contextual opinion can reflect the contextual uses (or conditions) of an item or a specific feature, which can be discovered from reviews through keyword matching (Chen and Chen 2014), rule-based reasoning (Li et al 2010), or a LDA-based classifier (Hariri et al 2011; Ramage et al 2009). In recommender systems, they can be combined with star ratings to infer a user’s utility of selecting an item in different contexts (Hariri et al 2011), or to model a user’s context-related latent factors (Li et al 2010) or context-dependent aspect preferences (Chen and Chen 2014; Levi et al 2012) (see Sections 4.3.4 and 4.4.2).

**6. Comparative opinions:** Another type of opinion that can be extracted from reviews is comparative opinion (Jindal and Liu 2006), such as the sentence “Bed was comfortable, perhaps not as good as some St. Regis’ but clearly better and more luxurious than the Westins heavenly stateside” (see Figure 2). Comparative opinions indicate whether an item is superior or inferior to another, with regard to some feature. Such opinions can be extracted using a set of special linguistic rules (Ganapathibhotla and Liu 2008). In Section 5.2, it can be seen that comparative opinions can be used to model products’ comparative relationships via a graph, and thus improve the products’ ranking quality (Jamroonsilp and Prompoon 2013;Li et al 2011; Zhang et al 2010).

**3. SYSTEM ANALYSIS**

**3.1 EXISTING SYSTEM**

Advanced text analysis and opinion mining techniques enable the extraction of various types of review elements, such as the discussed topics, the multi-faceted nature of opinions, contextual information, comparative opinions, and reviewers’emotions. The review-based recommender system’s ability to alleviate the well-known rating sparsity and cold-start problems is emphasized. This survey classifies state-of-the-art studies into two principal branches: review-based user profile building and review-based product profile building. In the user profile sub-branch, the reviews are not only used to create term-based profiles, but also to infer or enhance ratings. Multi-faceted opinions can further be exploited to derive the weight/value preferences that users place on particular features.

**3.2 PROPOSED SYSTEM**

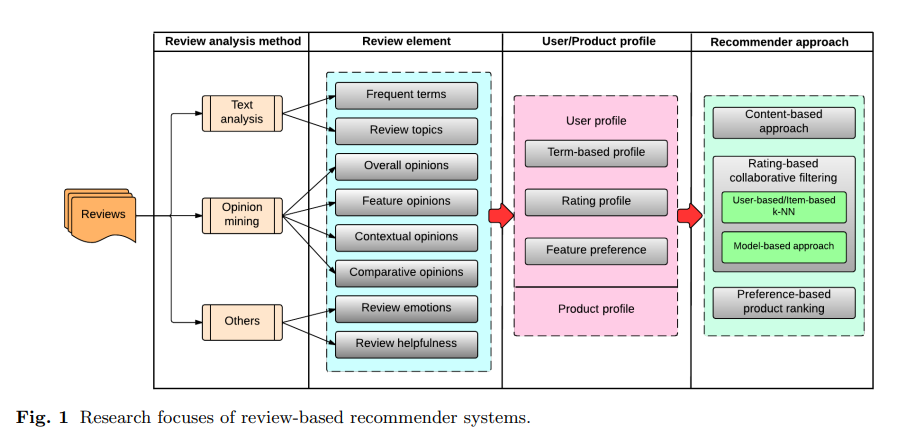
In this project we are recommending products to user by analysing past user’s AMAZON reviews data with the help of NLTK deep learning model. First we clean reviews and then extract ratings and reviews from dataset and then feed to NLTK deep learning algorithm to train a model. After training model application will accept product or brand name from user and then recommend new product to user based on reviews and ratings. This application will display rating and reviews also which describe why this recommended product is best.In this article, we provide a comprehensive overview of how the review elements have been exploited to improve standard content-based recommending, collaborative filtering, and preference-based product ranking techniques. In another sub-branch, the product profile can be enriched with feature opinions or comparative opinions to better reflect its assessment quality. The merit of each branch of work is discussed in terms of both algorithm development and the way in which the proposed algorithms are evaluated. In addition, we discuss several future trends based on the survey, which may inspire investigators to pursue additional studies in this area.

**MODUELS**

To implement this project we have designed following modules

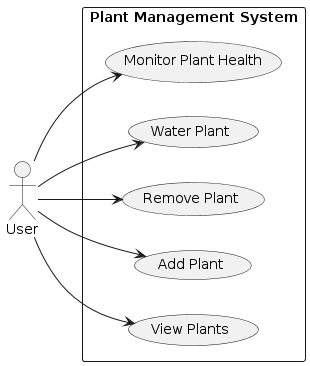
1. Register: using this module users can signup with application
2. Login: after signup user can login to application
3. Test Data for Recommendation: using this module user can enter any test data and if related data and its reviews available then application recommend best review product.

**SYSTEM ARCHITECTURE**

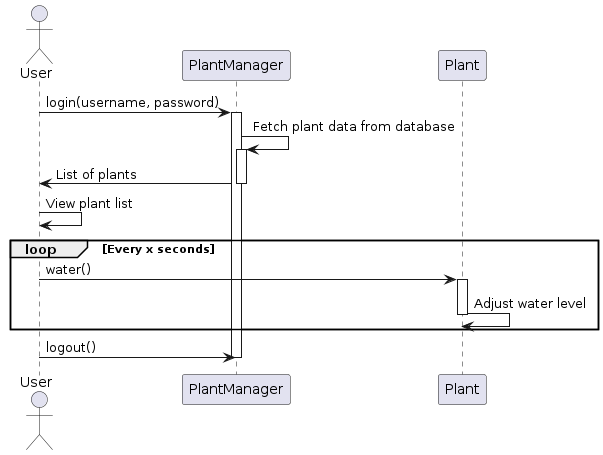


**SYSTEM DESIGN**

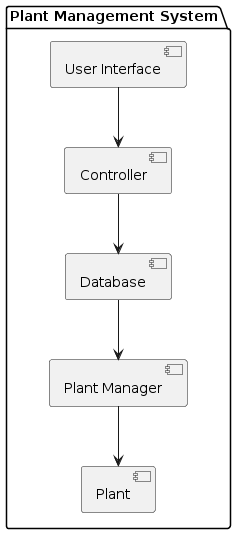
**Usecase Diagram:**



Sequence Diagram:



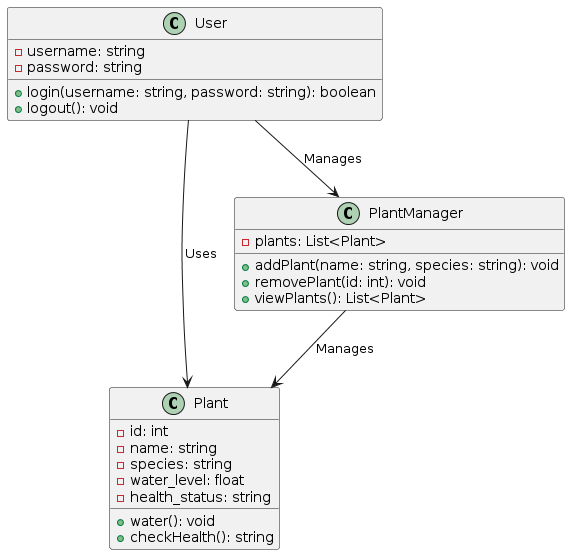
**Component Diagram:**



**Activity Diagram**



**Class Diagram:**



**SYSTEM STUDY**

**FEASIBILITY STUDY**

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

**Three key considerations involved in the feasibility analysis are,**

* **ECONOMICAL FEASIBILITY**
* **TECHNICAL FEASIBILITY**
* **SOCIAL FEASIBILITY**

**ECONOMICAL FEASIBILITY**

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

### TECHNICAL FEASIBILITY

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

**SOCIAL FEASIBILITY**

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

**FUNCTIONAL REQUIREMENTS**

In [software engineering](https://en.wikipedia.org/wiki/Software_engineering) and [systems engineering](https://en.wikipedia.org/wiki/Systems_engineering), a **functional requirement** defines a function of a [system](https://en.wikipedia.org/wiki/System) or its component, where a function is described as a specification of behavior between outputs and inputs.[[1]](https://en.wikipedia.org/wiki/Functional_requirement#cite_note-FultonAirborne17-1)

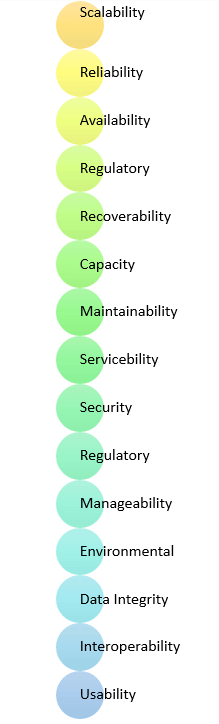
Functional requirements may involve calculations, technical details, data manipulation and processing, and other specific functionality that define what a system is supposed to accomplish.[[2]](https://en.wikipedia.org/wiki/Functional_requirement#cite_note-2) Behavioral requirements describe all the cases where the system uses the functional requirements, these are captured in [use cases](https://en.wikipedia.org/wiki/Use_case). Functional requirements are supported by [non-functional requirements](https://en.wikipedia.org/wiki/Non-functional_requirement) (also known as "quality requirements"), which impose constraints on the design or implementation (such as performance requirements, security, or reliability). Generally, functional requirements are expressed in the form "system must do <requirement>," while non-functional requirements take the form "system shall be <requirement>." The plan for implementing functional requirements is detailed in the system design, whereas *non-functional* requirements are detailed in the system architecture.[[4]](https://en.wikipedia.org/wiki/Functional_requirement#cite_note-AdamsNon15-4)[[5]](https://en.wikipedia.org/wiki/Functional_requirement#cite_note-J%C3%B6nssonImpact06-5)

As defined in [requirements engineering](https://en.wikipedia.org/wiki/Requirements_analysis), functional requirements specify particular results of a system. This should be contrasted with non-functional requirements, which specify overall characteristics such as cost and [reliability](https://en.wikipedia.org/wiki/Reliability_engineering). Functional requirements drive the application architecture of a system, while non-functional requirements drive the technical architecture of a system.[[4]](https://en.wikipedia.org/wiki/Functional_requirement#cite_note-AdamsNon15-4)

In some cases a requirements analyst generates use cases after gathering and validating a set of functional requirements. The hierarchy of functional requirements collection and change, broadly speaking, is: user/[stakeholder](https://en.wikipedia.org/wiki/Project_stakeholder) request → analyze → use case → incorporate. Stakeholders make a request; systems engineers attempt to discuss, observe, and understand the aspects of the requirement; use cases, entity relationship diagrams, and other models are built to validate the requirement; and, if documented and approved, the requirement is implemented/incorporated.[[6]](https://en.wikipedia.org/wiki/Functional_requirement#cite_note-MITRESys14-6) Each use case illustrates behavioral scenarios through one or more functional requirements. Often, though, an analyst will begin by eliciting a set of use cases, from which the analyst can derive the functional requirements that must be implemented to allow a user to perform each use case.

**NON-FUNCTIONAL REQUIREMENT** (NFR) specifies the quality attribute of a software system. They judge the software system based on Responsiveness, Usability, Security, Portability and other non-functional standards that are critical to the success of the software system. Example of nonfunctional requirement, *“how fast does the website load?”* Failing to meet non-functional requirements can result in systems that fail to satisfy user needs.

Non-functional Requirements allows you to impose constraints or restrictions on the design of the system across the various agile backlogs. Example, the site should load in 3 seconds when the number of simultaneous users are > 10000. Description of non-functional requirements is just as critical as a functional requirement.



* Usability requirement
* Serviceability requirement
* Manageability requirement
* Recoverability requirement
* Security requirement
* Data Integrity requirement
* Capacity requirement
* Availability requirement
* Scalability requirement
* Interoperability requirement
* Reliability requirement
* Maintainability requirement
* Regulatory requirement
* Environmental requirement

**Advantages of Non-Functional Requirement**

Benefits/pros of Non-functional testing are:

* The nonfunctional requirements ensure the software system follow legal and compliance rules.
* They ensure the reliability, availability, and performance of the software system
* They ensure good user experience and ease of operating the software.
* They help in formulating security policy of the software system.

**Disadvantages of Non-functional requirement**

Cons/drawbacks of Non-function requirement are:

* None functional requirement may affect the various high-level software subsystem
* They require special consideration during the software architecture/high-level design phase which increases costs.
* Their implementation does not usually map to the specific software sub-system,
* It is tough to modify non-functional once you pass the architecture phase.

**KEY LEARNING**

* A non-functional requirement defines the performance attribute of a software system.
* Types of Non-functional requirement are Scalability Capacity, Availability, Reliability, Recoverability, Data Integrity, etc.
* Example of Non Functional Requirement is Employees never allowed to update their salary information. Such attempt should be reported to the security administrator.
* Functional Requirement is a verb while Non-Functional Requirement is an attribute
* The advantage of Non-functional requirement is that it helps you to ensure good user experience and ease of operating the software
* The biggest disadvantage of Non-functional requirement is that it may affect the various high-level software subsystems.

**CODING**

from django.shortcuts import render

from django.template import RequestContext

from django.contrib import messages

import pymysql

from django.http import HttpResponse

import pandas as pd

import numpy as np

import re

from sklearn.feature\_extraction.text import TfidfVectorizer

from numpy import dot

from numpy.linalg import norm

from vaderSentiment.vaderSentiment import SentimentIntensityAnalyzer

from nltk.corpus import stopwords

stop\_words = set(stopwords.words('english'))

sid = SentimentIntensityAnalyzer()

def getReview(review):

review\_result = "none"

review = review.lower()

review = re.sub('[^A-Za-z]+', ' ', review)

sentiment\_dict = sid.polarity\_scores(review.strip())

compound = sentiment\_dict['compound']

if compound >= 0.05 :

review\_result = 'Positive'

return review\_result

dataset = pd.read\_csv("Dataset/amazon\_reviews.csv")

dataset = dataset.values

text = dataset[:,0]

label = dataset[:,1]

tfidf\_vectorizer = TfidfVectorizer(stop\_words=stop\_words, use\_idf=True, smooth\_idf=False, norm=None, decode\_error='replace', max\_features=1000,lowercase=True)

tfidf = tfidf\_vectorizer.fit\_transform(text).toarray()

df = pd.DataFrame(tfidf, columns=tfidf\_vectorizer.get\_feature\_names())

print(df.shape)

df = df.values

X = df[:, 0:1000]

def index(request):

if request.method == 'GET':

return render(request, 'index.html', {})

def Login(request):

if request.method == 'GET':

return render(request, 'Login.html', {})

def Register(request):

if request.method == 'GET':

return render(request, 'Register.html', {})

def Recommendation(request):

if request.method == 'GET':

return render(request, 'Recommendation.html', {})

def RecommendationAction(request):

if request.method == 'POST':

query = request.POST.get('t1', False)

test = query.lower().strip()

test = tfidf\_vectorizer.transform([test]).toarray()

test = test[0]

similarity = 0

review = 'Unable to get review for recommendation'

rating = 0

suggestion = "No suggestion available"

for j in range(len(X)):

review\_score = dot(X[j], test)/(norm(X[j])\*norm(test))

if review\_score > similarity:

similarity = review\_score

review\_type = getReview(text[j])

if review\_type == 'Positive':

review = text[j]

rating = label[j]

suggestion = "you have chosen best product"

output="<html><body><center><table border=1><tr><th><font size=3 color=black>Product Name</th>"

output+="<th><font size=3 color=black>Recommended Best Review</th>"

output+="<th><font size=3 color=black>Recommended Best Rating</th><th><font size=3 color=black>Suggestion</th></tr>"

output+="<tr><td><font size=3 color=black>"+query+"</td><td><font size=3 color=black>"+review+"</td><td><font size=3 color=black>"+str(rating)+"</td><td><font size=3 color=black>"+suggestion+"</td>"

#output+"</tr><br/><br/><br/><br/><br/><br/></table>"

context= {'data':output}

return render(request, 'Result.html', context)

def Signup(request):

if request.method == 'POST':

#user\_ip = getClientIP(request)

#reader = geoip2.database.Reader('C:/Python/PlantDisease/GeoLite2-City.mmdb')

#response = reader.city('103.48.68.11')

#print(user\_ip)

#print(response.location.latitude)

#print(response.location.longitude)

username = request.POST.get('username', False)

password = request.POST.get('password', False)

contact = request.POST.get('contact', False)

email = request.POST.get('email', False)

address = request.POST.get('address', False)

db\_connection = pymysql.connect(host='127.0.0.1',port = 3306,user = 'root', password = 'root', database = 'Recommendation',charset='utf8')

db\_cursor = db\_connection.cursor()

student\_sql\_query = "INSERT INTO register(username,password,contact,email,address) VALUES('"+username+"','"+password+"','"+contact+"','"+email+"','"+address+"')"

db\_cursor.execute(student\_sql\_query)

db\_connection.commit()

print(db\_cursor.rowcount, "Record Inserted")

if db\_cursor.rowcount == 1:

context= {'data':'Signup Process Completed'}

return render(request, 'Register.html', context)

else:

context= {'data':'Error in signup process'}

return render(request, 'Register.html', context)

def UserLogin(request):

if request.method == 'POST':

username = request.POST.get('username', False)

password = request.POST.get('password', False)

utype = 'none'

con = pymysql.connect(host='127.0.0.1',port = 3306,user = 'root', password = 'root', database = 'Recommendation',charset='utf8')

with con:

cur = con.cursor()

cur.execute("select \* FROM register")

rows = cur.fetchall()

for row in rows:

if row[0] == username and row[1] == password:

utype = 'success'

break

if utype == 'success':

file = open('session.txt','w')

file.write(username)

file.close()

context= {'data':'welcome '+username}

return render(request, 'UserScreen.html', context)

if utype == 'none':

context= {'data':'Invalid login details'}

return render(request, 'Login.html', context)

**SYSTEM TEST**

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub assemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

### TYPES OF TESTS

**Unit testing**

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

**Integration testing**

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

**Functional test**

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

Valid Input : identified classes of valid input must be accepted.

Invalid Input : identified classes of invalid input must be rejected.

Functions : identified functions must be exercised.

Output : identified classes of application outputs must be exercised.

Systems/Procedures : interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

**System Test**

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

**White Box Testing**

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.

**Black Box Testing**

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box .you cannot “see” into it. The test provides inputs and responds to outputs without considering how the software works.

**Unit Testing**

Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

**Test strategy and approach**

Field testing will be performed manually and functional tests will be written in detail.

**Test objectives**

* All field entries must work properly.
* Pages must be activated from the identified link.
* The entry screen, messages and responses must not be delayed.

**Features to be tested**

* Verify that the entries are of the correct format
* No duplicate entries should be allowed
* All links should take the user to the correct page.

# Integration Testing

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.

The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

**Test Results:** All the test cases mentioned above passed successfully. No defects encountered.

**Acceptance Testing**

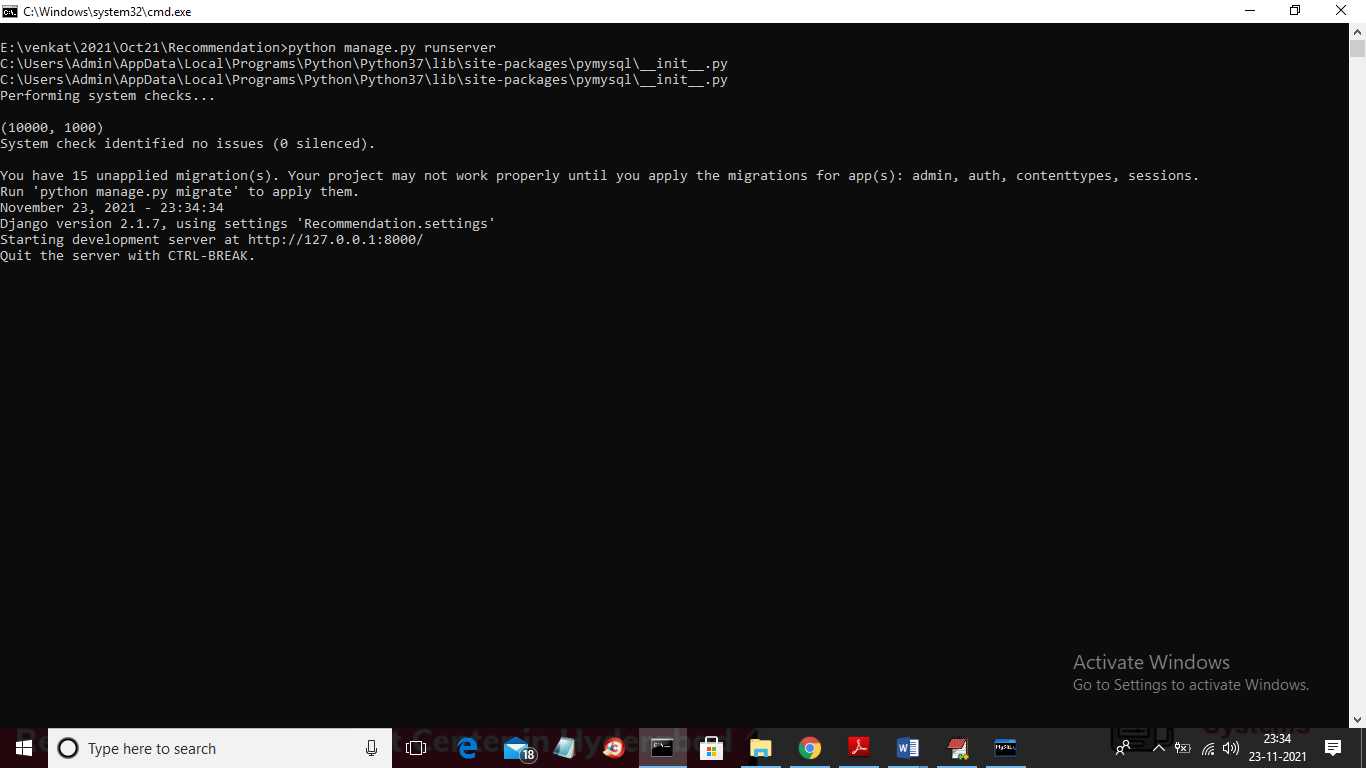
User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

**Test Results:** All the test cases mentioned above passed successfully. No defects encountered.

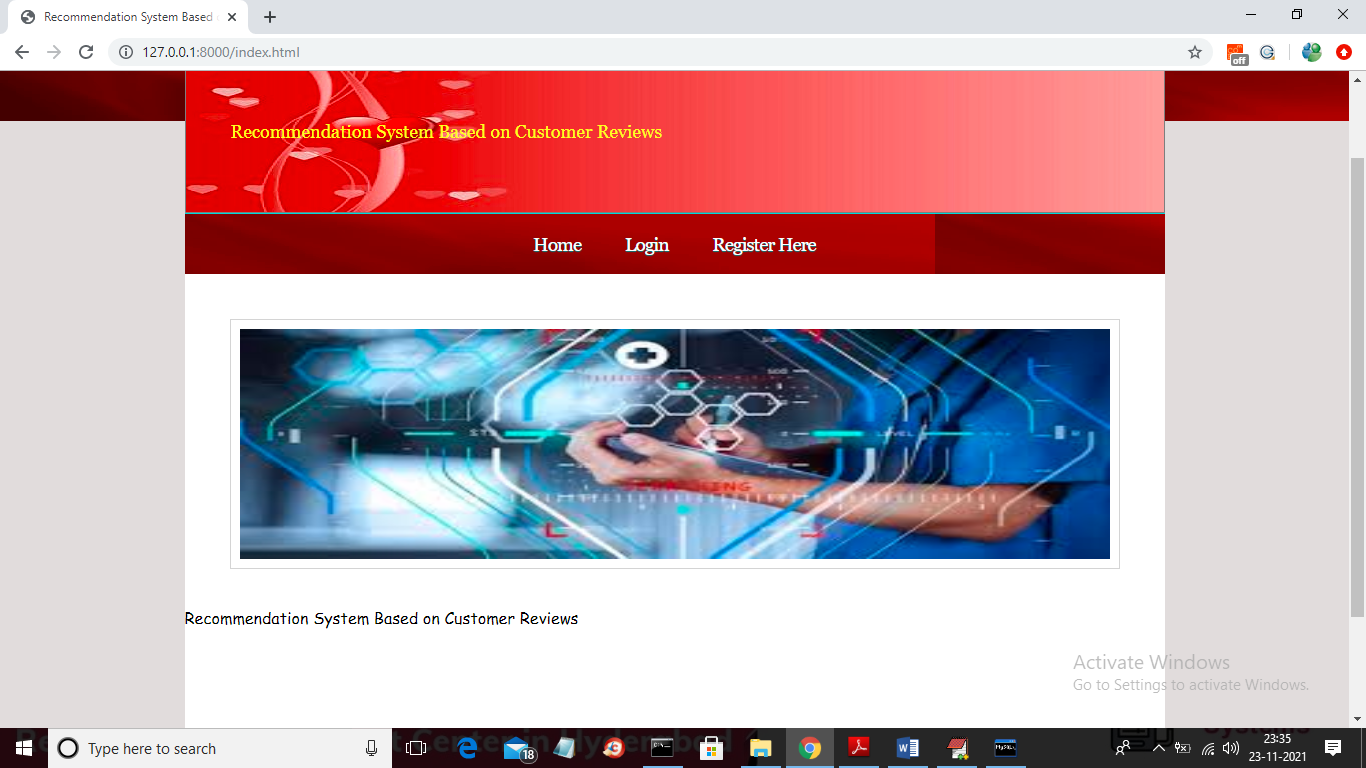
**SCREENS SHOTS**

To run project first create database in MYSQL by copying content from ‘DB.txt’ file and then paste in MYSQL

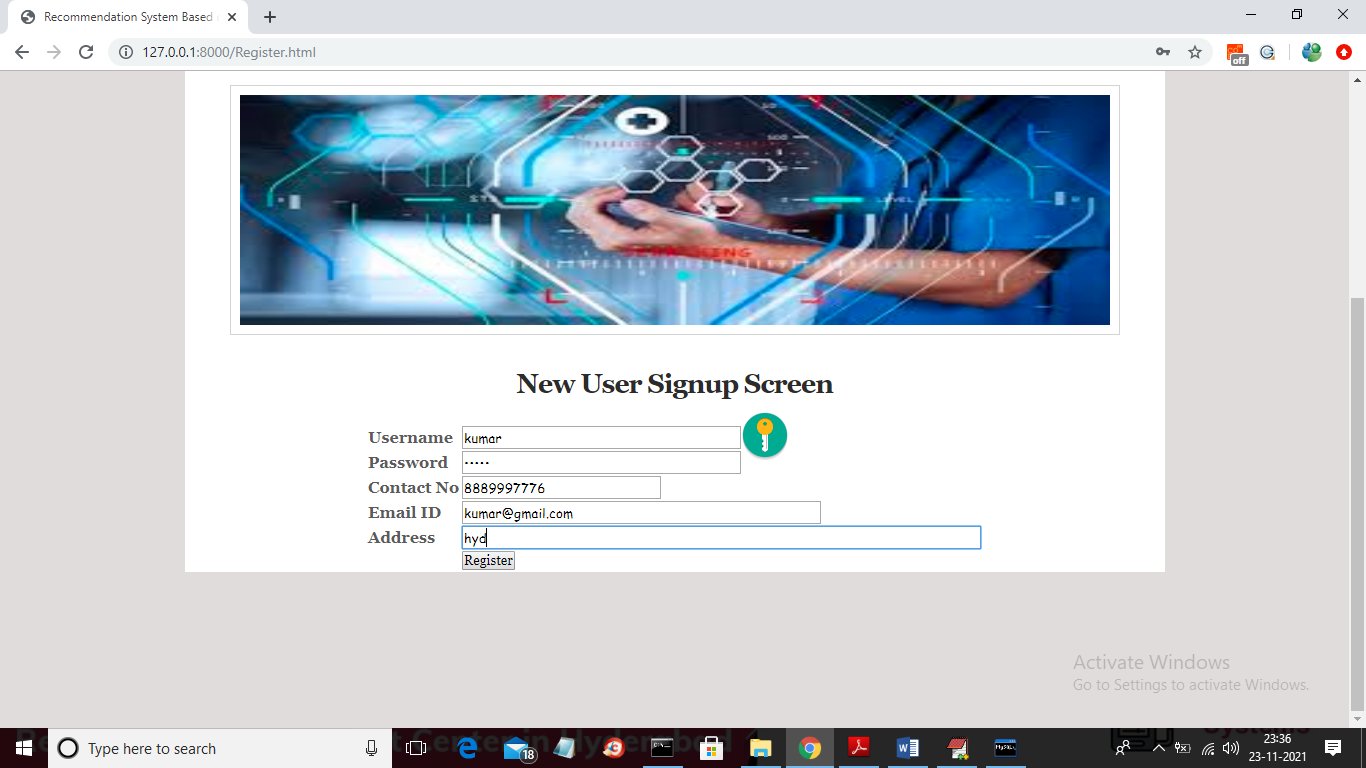
Now double click on ‘runServer.bat’ file to start DJANGO webserver like below screen



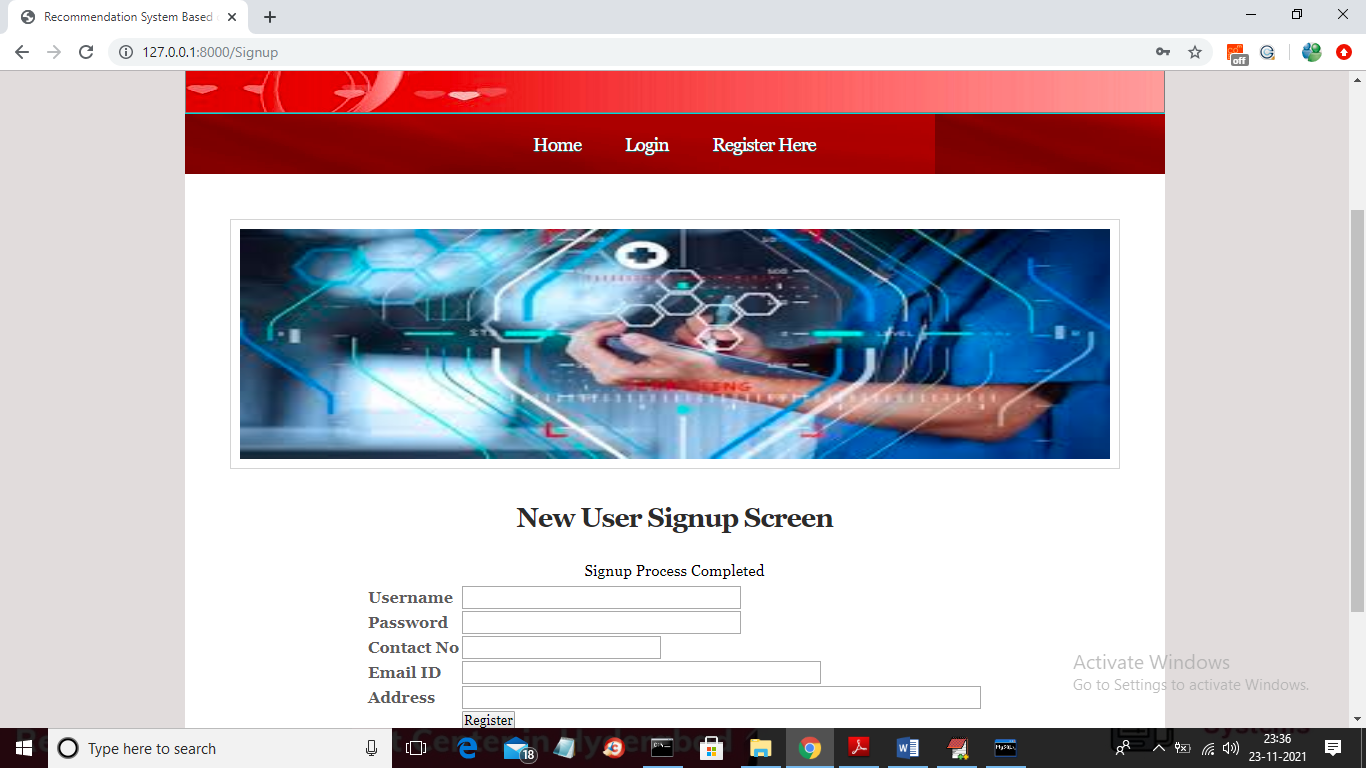
In above screen server started and now open browser and enter URL as ‘http://127.0.0.1:8000/index.html’ and press enter key to get below screen



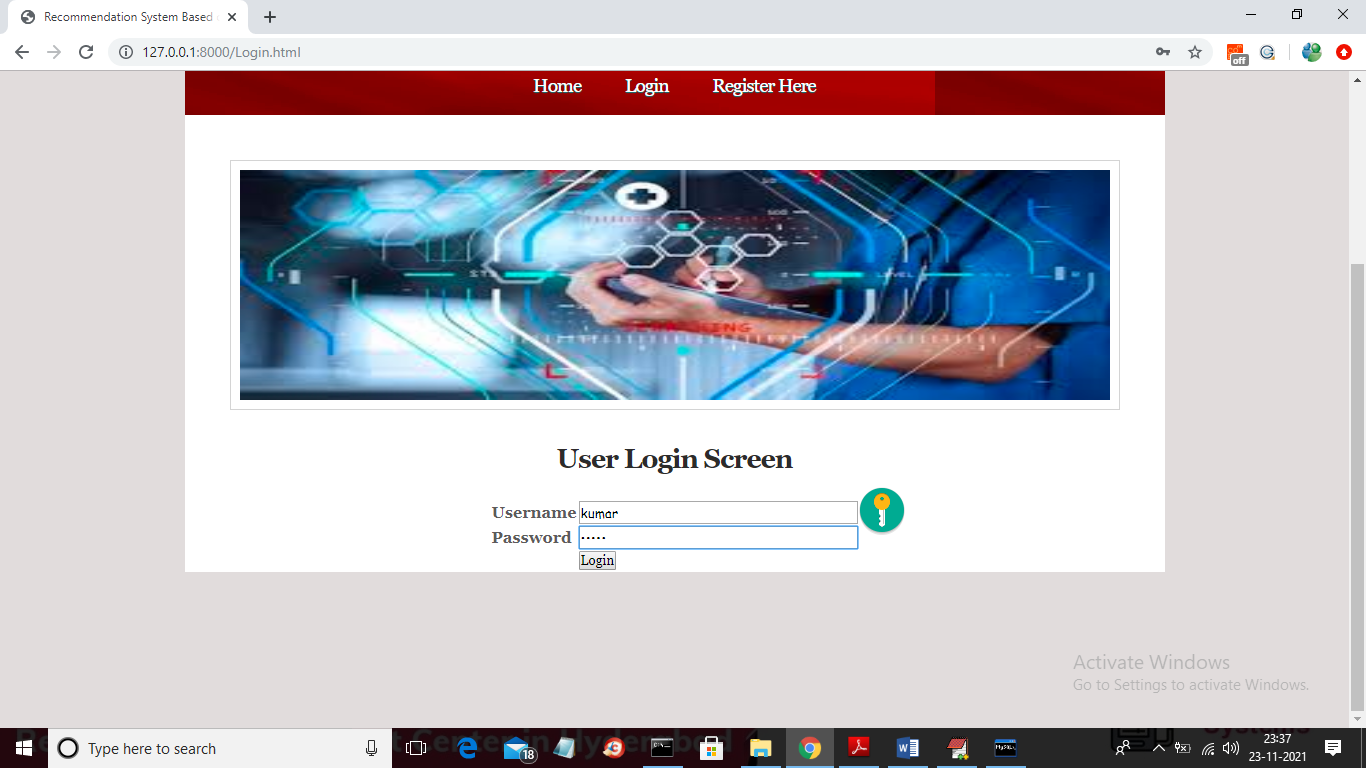
In above screen click on ‘Register Here’ link to get below signup page



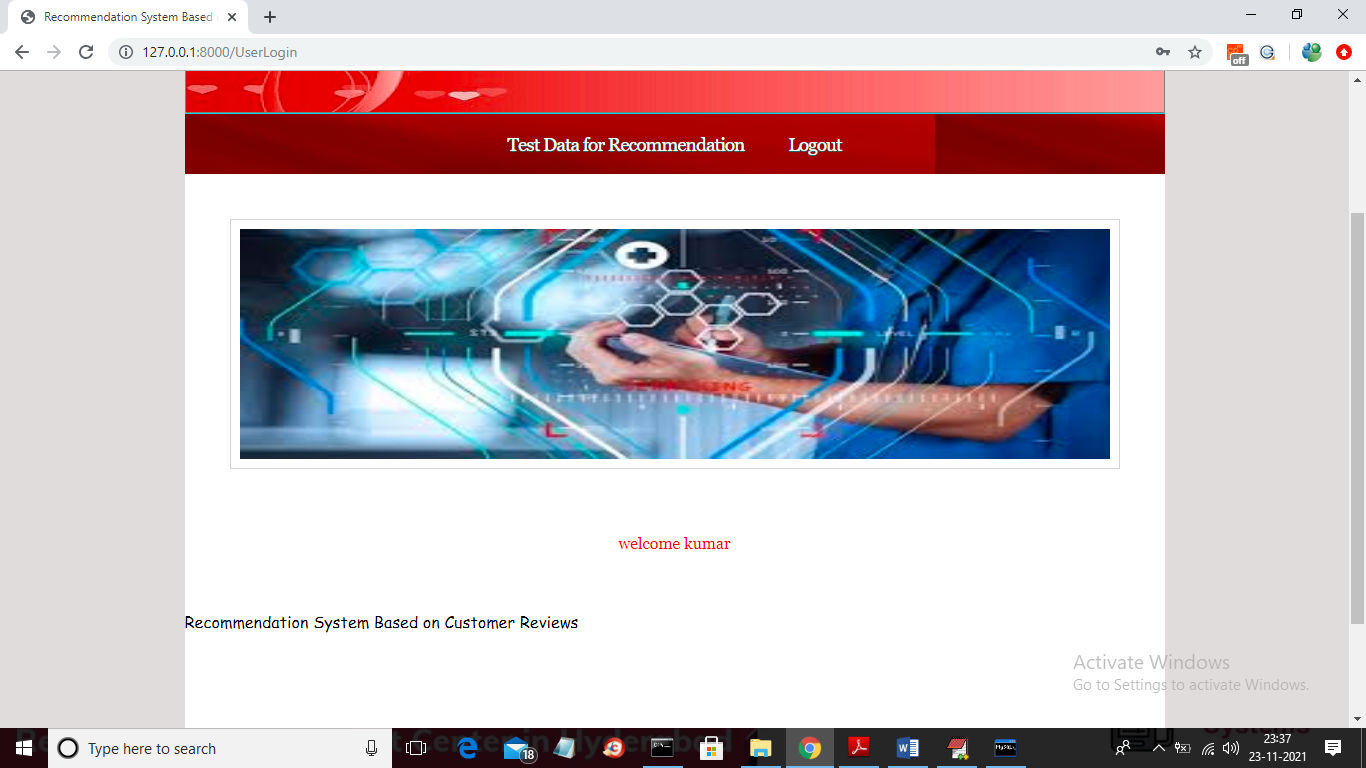
In above screen user entering signup details and then click on ‘Register’ button to get below screen



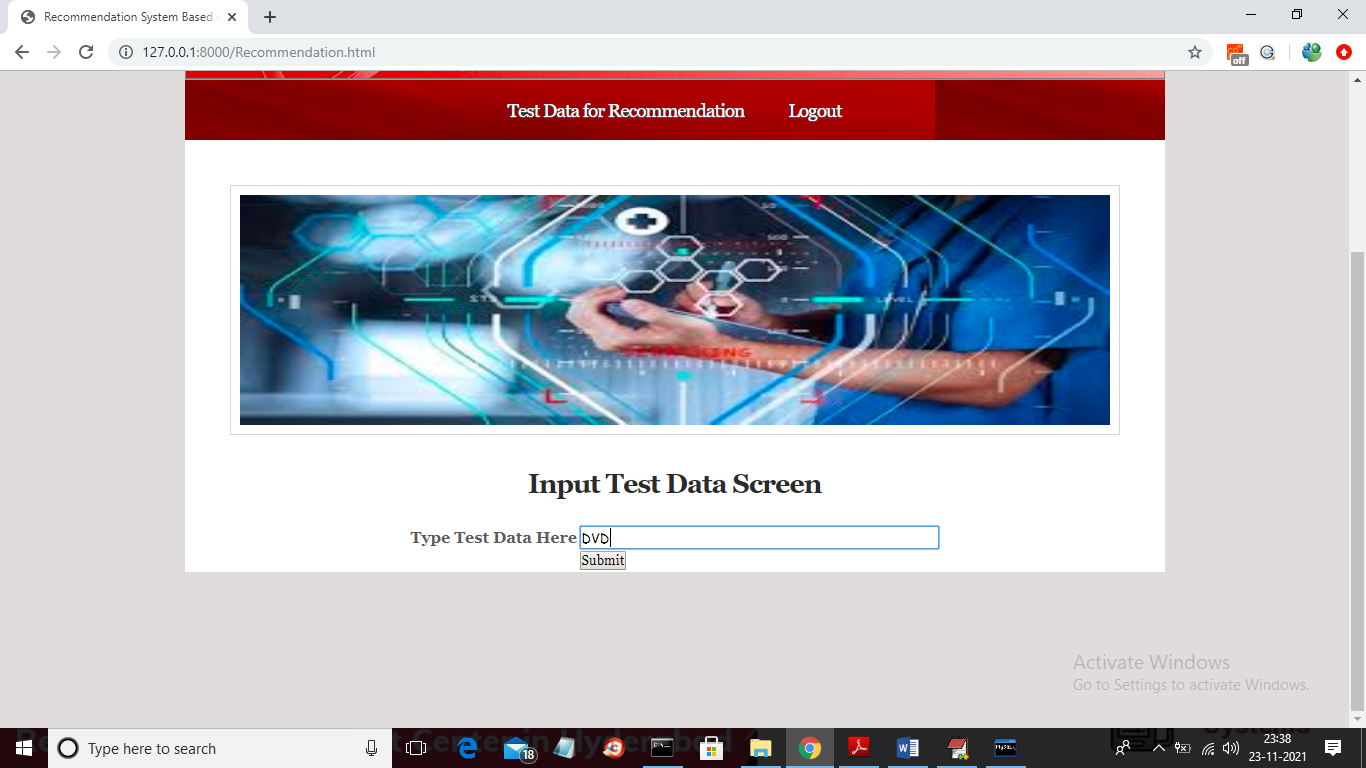
In above screen signup process completed and now click on ‘Login’ link to get below screen



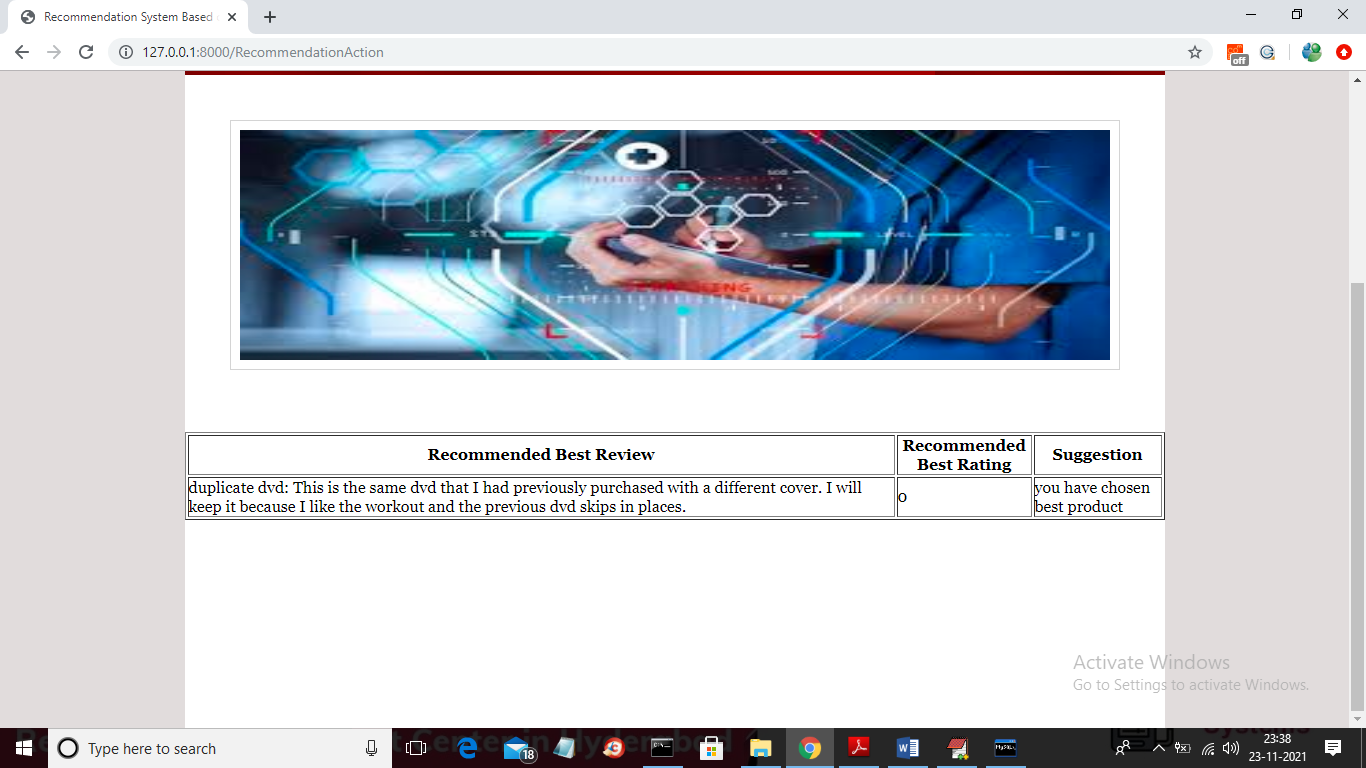
In above screen user is login and after login will get below screen



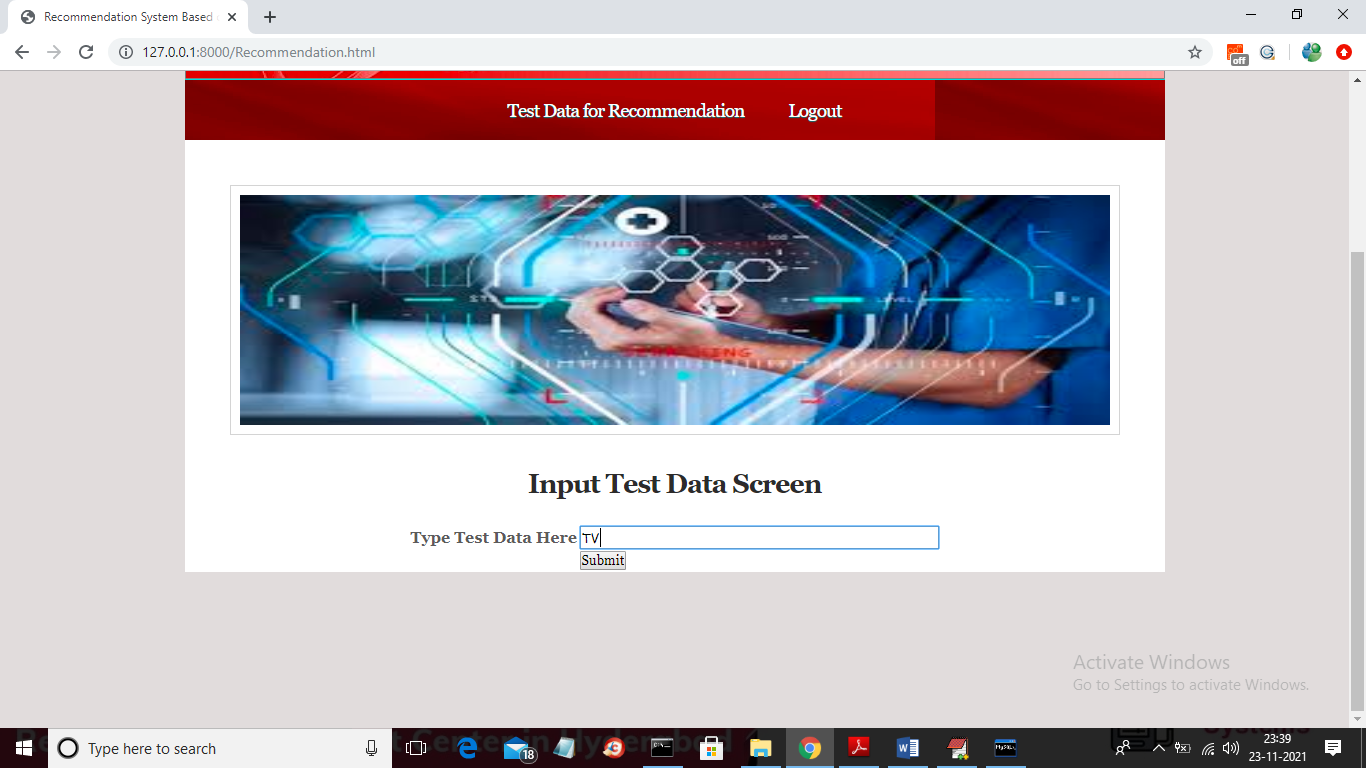
In above screen click on ‘Test Data for Recommendation’ link to get below screen



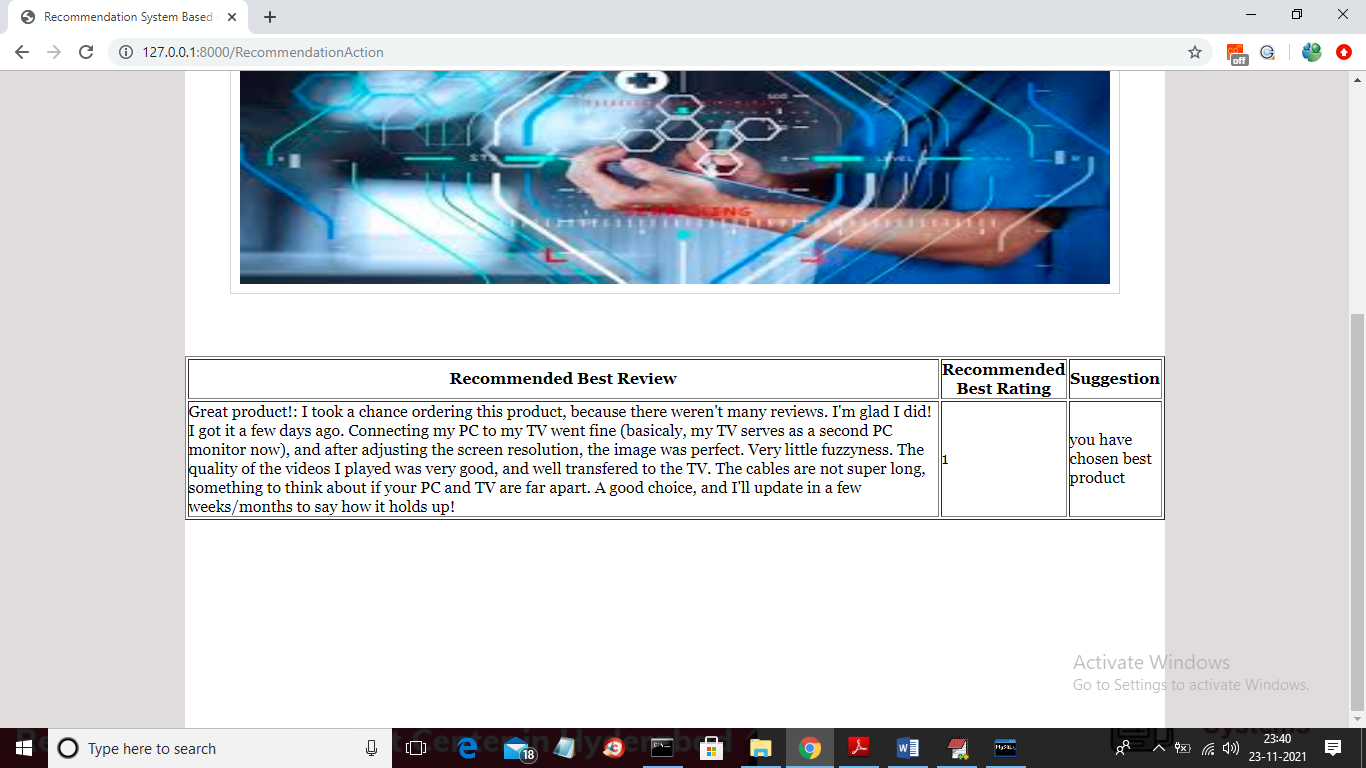
In above screen I entered product name as ‘DVD’ and press button to get below output



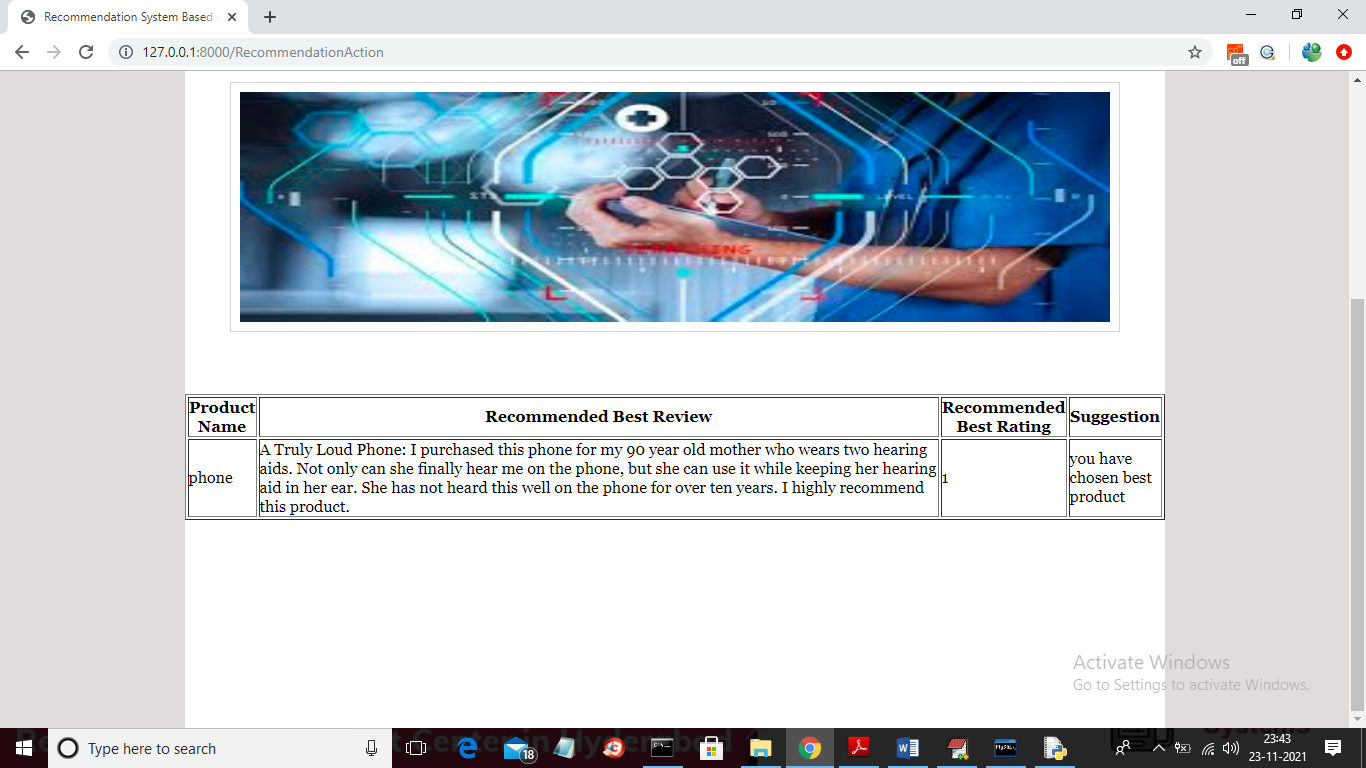
In above screen we are getting recommended review and rating for given product and in above 0 means rating is low and 1 means rating is high and now try another product



In above screen I entered product as TV and press button to get recommended review



In above screen we got rating as 1 which means product is recommended and similarly you can give any test data from dataset to get recommended reviews and ratings



**CONCLUSION:**

In this article, we survey state-of-the-art research on review-based recommender systems.We classify the systems according to the two main types of profile building: review-based user profile building, and review-based product profile building. For the first category, we discuss how existing studies have used reviews to create term-based user profile, enrich rating profile, and derive feature preference. Various types of review elements, such as review helpfulness, review topics, overall opinions, feature opinions, review contexts, and review emotions, have been used to enhance the standard content-based recommending method and rating-based collaborative filtering method. In the category of product profile building, feature opinions and comparative opinions have been exploited, which can be helpful for increasing the products’ ranking accuracy.We further discuss the practical implications of these studies in terms of solving the well-known rating sparsity and new user problems, and their proven ability to improve the currently used algorithms and practical uses in different types of product domains.We expect this survey to encourage investigators to pursue the hidden values of reviews in future studies. For instance, combining multiple types of review elements might be more effective than considering a single type when modeling a user’s preference. The effects of reviews on enhancing multi-criteria recommenders, context-aware recommenders, and emotion-based recommenders could be investigated in more comprehensive studies. More realistic evaluation techniques, such as user evaluation, could validate the practical benefits of the review-based recommending method. Beyond recommendation, reviews could also be exploited to design more effective user interfaces, such as an explanation interface.

REFERENCES

1. Acar E, Dunlavy DM, Kolda TG, Mørup M (2011) Scalable tensor factorizations for incomplete data. Chemometrics and Intelligent Laboratory Systems 106(1):41–56 Aciar S, Zhang D, Simoff S, Debenham J (2007) Informed recommender: Basing recommendations on consumer product reviews. IEEE Intelligent Systems 22(3):39 –47 Adomavicius G, Kwon Y (2007) New recommendation techniques for multicriteria rating systems.
2. IEEE Intelligent Systems 22(3):48–55 Adomavicius G, Tuzhilin A (2005) Toward the next generation of recommender systems: A survey of the stateof-the-art and possible extensions. IEEE Transactions on Knowledge and Data Engineering 17(6):734–749 Adomavicius G, Tuzhilin A (2011) Context-aware recommender systems.
3. In: Ricci F, Rokach L, Shapira B, Kantor PB (eds) Recommender Systems Handbook, Springer, pp 217–253 Al-Taie MZ (2013) Explanations in recommender systems: Overview and research approaches. In: Proceedings of the 14th International Arab Conference on Information Technology, Khartoum, Sudan, ACIT’13.
4. Bach F, Jordan MI (2005) A probabilistic interpretation of canonical correlation analysis. Tech. Rep. 688, Department of Statistics, University of California, Berkeley, USA
5. Balabanovi´c M, Shoham Y (1997) Fab: Content-based, collaborative recommendation. Communications of the ACM 40(3):66–72
6. Baltrunas L, Ludwig B, Peer S, Ricci F (2012) Context relevance assessment and exploitation in mobile recommender systems. Personal and Ubiquitous Computing 16(5):507–526
7. Beilin L, Yi S (2013) Survey of personalized recommendation based on society networks analysis. In: Proceedings of the 6th International Conference on Information Management, Innovation Management and Industrial Engineering, Xi’an, China, ICIII’ 13, vol 3, pp 337–340
8. Blei DM, Ng AY, Jordan MI (2003) Latent dirichlet allocation. Journal of Machine Learning Research 3:993–1022 Celma O, Herrera P (2008) A new approach to evaluating novel recommendations. In: Proceedings of the 2nd ACM International Conference on Recommender Systems, Lausanne, Switzerland, ACM, RecSys’08, pp 179–186
9. Chatterjee P (2001) Online reviews: Do consumers use them? Advances in Consumer Research 28:129–133 Chee SHS, Han J, Wang K (2001) Rectree: An efficient collaborative filtering method. In: Kambayashi Y, Winiwarter W, Arikawa M (eds) Proceedings of the 3rd International Conference on Data Warehousing and Knowledge Discovery, Munich, Germany, Springer-Verlag, DaWaK’01, pp 141–151
10. Chelcea S, Gallais G, Trousse B (2004) A personalized recommender system for travel information. In: Proceedings of the 1st French-speaking Conference on Mobility and Ubiquity Computing, Nice, France, ACM, UbiMob’04,pp 143–150
11. Chen G, Chen L (2014) Recommendation based on contextual opinions. In: Dimitrova V, Kuflik T, Chin D,Ricci F, Dolog P, Houben GJ (eds) Proceedings of the 22nd Conference on User Modeling, Adaptation, and Personalization, Alborg, Denmark, Springer, UMAP’14, pp 61–73
12. Chen L, Pu P (2004) Survey of preference elicitation methods. Tech. Rep. IC/200467, Swiss Federal Institute of Technology in Lausanne (EPFL), Lausanne, Switzerland Chen L, Pu P (2010) Experiments on the preference-based organization interface in recommender systems. ACM Transactions on Computer-Human Interaction 17(1):5:1–5:33
13. Chen L, Wang F (2013) Preference-based clustering reviews for augmenting e-commerce recommendation.Knowledge-Based Systems 50:44–59 Chen L, Wang F (2014) Sentiment-enhanced explanation of product recommendations. In: Proceedings of the 23rd International Conference on World Wide Web Companion, Seoul, Korea, ACM, WWW Companion’14,pp 239–240
14. Chen L, Zeng W, Yuan Q (2013) A unified framework for recommending items, groups and friends in social media environment via mutual resource fusion. Expert Systems with Applications 40(8):2889–2903
15. Chevalier JA, Mayzlin D (2006) The effect of word of mouth on sales: Online book reviews. Journal of Marketing Research 43(3):345–354
16. Dong R, O’Mahony MP, Schaal M, McCarthy K, Smyth B (2013a) Sentimental product recommendation. In:Proceedings of the 7th ACM International Conference on Recommender Systems, Hong Kong, China, ACM,RecSys’13, pp 411–414.