**Rating Prediction based on Social Sentiment Analysis from Textual Reviews**

Project report submitted to

*In partial fulfillment of the requirements*

*For the award of the degree*

*Of*

**BACHELOR OF TECHNOLOGY**

**IN**

**COMPUTER SCIENCE AND ENGINEERING**

**By**

**ARAVIND D N MOHAN ADITYA S BHARGAV S HARSHITH**

**(1210314416) (1210314441) (1210314451) (1210314459)**

Under the esteemed guidance of

**Dr. K. Subhadra**

**Assistant Professor**



**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**GITAM INSTITUTE OF TECHNOLOGY**

**GITAM Deemed to be UNIVERSITY**

**RUSHIKONDA, VISHAKHAPATNAM-530045**

**DEPT. OF COMPUTER SCIENCE AND ENGINNERING**

**GITAM**

**Deemed to be University**

**VISAKHAPATNAM, A.P, INDIA**



**CERTIFICATE**

This is to certify that this project report entitled “**RATING PREDICTION BASED ON SOCIAL SENTIMENT ANALYSIS FROM TEXTUAL REVIEWS**” is a bonafide work done by ARAVIND D, N MOHAN ADITYA, S BHARGAV, S HARSHITH in the partial fulfillment of the requirements for the award of **Bachelor of Technology in Computer Science and Engineering** at GITAM Deemed to be University, Visakhapatnam.

PROJECT GUIDE HEAD OF THE DEPARTMENT

**Dr. K. Subhadra Dr. K. Thammi Reddy Assistant Professor Professor Dept. of CSE Dept. of CSE**

**DECLARATION**

We, (ARAVIND D, N MOHAN ADITYA, S BHARGAV, S HARSHITH) hereby declare that the project report entitled “**RATING PREDICTION BASED ON SOCIAL SENTIMENT ANALYSIS FROM TEXTUAL REVIEWS**” is an original and authentic work done in the Dept. of CSE, GITAM Institute of Technology, GITAM Deemed to be UNIVERSITY, Rushikonda, Vishakhapatnam submitted in partial fulfillment of requirements for the award of the degree of Bachelor of Technology in Computer Science and Engineering. The matter embodied in this project work has not been submitted earlier for the award of any degree or diploma to the best of our knowledge.

**ARAVIND D N MOHAN ADITYA S BHARGAV S HARSHITH**

**(1210314416) (1210314441) (1210314451) (1210314459)**

**ACKNOWLEDGEMENT**

The satisfaction that accompanies the successful completion of any task would be incomplete without the mention of the people who made it possible and whose constant guidance and encouragement crown all the efforts with the success.

We would like to express our deep sense of gratitude and sincere thanks to our project guide **Dr. K. Subhadra,** Assistant Professor,Dept. of Computer Science and Engineering, for her constant guidance, supervision and motivation in completing the project.

We are extremely elated to extend our gratitude to **Dr. K. Thammi Reddy,** Professor and Head of the Department, Dept. of Computer Science and Engineering, for his encouragement all the way during this project. His annotations, insinuations and criticisms are the key behind the successful completion of the project.

We gratefully acknowledge **Prof. Lakshmi Prasad,** Principal, GITAM Institute of Technology, GITAM Deemed to be UNIVERSITY, for providing necessary infrastructure and resources to accomplish my research work.

We express our sincere thanks to Teaching and Non-teaching staff members of the GITAM Institute of Technology, GITAM Deemed to be UNIVERSITY, for their constant support during the work.

**ABSTRACT**

The eagerness of human being to know about item or product before he or she going to buy any particular item which is available in online markets inspire our team to work on this project. In traditional review systems, we have witnessed a flourish of review websites. This project presents a great opportunity to share our viewpoints for various products we purchase. How to mine valuable information from reviews to understand a user’s preferences and make an accurate recommendation is crucial. Traditional recommender systems (RS) consider some factors, such as user’s purchase records, product category, and geographic location.

Sentiment analysis refers to the use of natural language processing, text analysis, computational linguistics, and biometrics to systematically identify, extract, quantify, and study affective states and subjective information. It is widely applied to voice of the customer materials such as reviews and survey responses, online and social media, and healthcare materials for applications that range from marketing to customer service to clinical medicine. It aims to determine the attitude of a speaker, writer, or other subject with respect to some topic or the overall contextual polarity or emotional reaction to a document, interaction, or event. The attitude may be a judgment or evaluation (see appraisal theory), affective state (that is to say, the emotional state of the author or speaker), or the intended emotional communication.

Firstly, we took a proposal of social user sentimental measurement approach and calculate each user’s sentiment on items/products. Secondly, we not only consider a user’s own sentimental attributes but also taking interpersonal sentimental influence into consideration. Then, we consider product reputation, which can be inferred by the sentimental distributions of a user set that reflect customers’ comprehensive evaluation. At last, we fuse three factors—user sentiment similarity, interpersonal sentimental influence, and item’s reputation similarity—into our recommender system to make an accurate rating prediction. We conduct a performance evaluation of the three sentimental factors on a real-world dataset collected from Yelp. Our experimental results show the sentiment can well characterize user preferences, which help to improve the recommendation performance.

**INDEX**

**1. Introduction and Motivation**

1.1 Introduction…………………………………………………………………………1

1.2 Existing System……………………………………………………………………..3

1.3 Proposed System……………………………………………………………………4

**2. Literature Review ……………………………………………………………………..5**

**3. Analysis**

3.1 Scope……………………………………………………………………………….6

3.2 Objectives…………………………………………………………………………..6

3.3 Requirements……………………………………………………………………….7

3.3.1 Functional Requirements

3.3.2 Non-Functional Requirements

3.3.2.1 Hardware Requirements

3.3.2.2 Software Requirements

**4. Design………………………………………………………………………………….12**

**5. Sample Code…………………………………………………………………………..41**

**6. Testing…………………………………………………………………………………48**

**7. Results…………………………………………………………………………………52**

**8. Conclusion and Future Work………………………………………………………..55**

**9. References……………………………………………………………………………..56**

**List of Figures**

Fig 1: Use Case Diagram…………………………………………………………………13

Fig 2: Class Diagram……………………………………………………………………..14

Fig 3: Sequence Diagram………………………………………………………………...15

Fig 4: Activity Diagram………………………………………………………………….17

Fig 5: Data Flow Diagram………………………………………………………………..19

Fig 6: System Architecture……………………………………………………………….20

Fig 7: Java Virtual Machine……………………………………………………………...24

Fig 8: Java Technology…………………………………………………………………..24

Fig 9: Java Platform………………………………………………………………………25

Fig 10: Java Architecture…………………………………………………………………27

Fig 11: Java Flow Diagram……………………………………………………………….32

Fig 12: TCP/IP Stack...……………………………………………………………………33

Fig 13: IP Address………………………………………………………………………...35

Fig 14: J2ME Architecture………………………………………………………………..37

Fig 15: Home Page………………………………………………………………………..52

Fig 16: All Product Items………………………………………………………………….52

Fig 17: Overall User Ratings………………………………………………………………53

Fig 18: Product items Separation………………………………………………………….53

Fig 19: Item based User Ratings Separation………………………………………………54

Fig 20: All Product Categories…………………………………………………………….54

**List of Tables**

Table 1: Test Cases…………………………………………………………………………51

**1. INTRODUCTION and MOTIVATION**

**1.1 Introduction**

There is much personal information in online textual reviews, which plays a very important role on decision processes. For example, the customer will decide what to buy if he or she sees valuable reviews posted by others, especially user’s trusted friend. We believe reviews and reviewers will do help to the rating prediction based on the idea that high-star ratings may greatly be attached with good reviews. Hence, how to mine reviews and the relation between reviewers in social networks has become an important issue in web mining, machine learning and natural language processing.

We focus on the rating prediction task. However, user’s rating star-level information is not always available on many review websites. Conversely, reviews contain enough detailed product information and user opinion information, which have great reference value for a user’s decision. Most important of all, a given user on website is not possible to rate every item. Hence, there are many unrated items in a user-item-rating matrix. It is inevitable in many rating prediction approaches. Review/comment, as we all know, is always available. In such case, it’s convenient and necessary to leverage user reviews to help predicting the unrated items.

The rise like DouBan1, Yelp2 and other review websites provides a broad thought in mining user preferences and predicting user’s ratings. Generally, user’s interest is stable in short term, so user topics from reviews can be representative. For example, in the category of Cups & Mugs, different people have different tastes. Some people pay attention to the quality, some people focus on the price and others may evaluate comprehensively. Whatever, they all have their personalized topics. Most topic models introduce users’ interests as topic distributions according to reviews contents. They are widely applied in sentiment analysis, travel recommendation, and social networks analysis.

Sentiment analysis is the most fundamental and important work in extracting user’s interest preferences. In general, sentiment is used to describe user’s own attitude on items. We observe that in many practical cases, it is more important to provide numerical scores rather than binary decisions. Generally, reviews are divided into two groups, positive and negative. However, it is difficult for customers to make a choice when all candidate products reflect positive sentiment or negative sentiment. To make a purchase decision, customers not only need to know whether the product is good, but also need to know how good the product is. It’s also agreed that different people may have different sentimental expression preferences. For example, some users prefer to use “good” to describe an “excellent” product, while others may prefer to use “good” to describe a “just so” product.

In our daily life, customers are most likely to buy those products with highly-praised reviews. That is, customers are more concerned about item’s reputation, which reflects consumers’ comprehensive evaluation based on the intrinsic value of a specific product. To obtain the reputation of a product, sentiment in reviews is necessary. Normally, if item’s reviews reflect positive sentiment, the item may be with good reputation to a great extent. Oppositely, if item’s reviews are full of negative sentiment, then the item is to be with bad reputation. To a given product, if we know user sentiment, we can infer the reputation and even the comprehensive ratings. When we search the net for purchasing, both positive reviews and negative reviews are valuable to be as reference. For positive reviews, we can know the advantages of a product. For negative reviews, we can obtain the shortcomings in case of being cheated. So it’s worth to explore those reviewers who have obvious and objective attitude on items. We observe that reviewers’ sentiment will influence others: if a reviewer has clear like and dislike sentiment, other users will pay much attention to him/her. However, user’s sentiment is hard to predict and the unpredictability of interpersonal sentimental influence makes a great difficulty in exploring social users.

**1.2 Existing System**

* Sentiment analysis can be conducted on three different levels: review-level, sentence-level, and phrase-level.
* Review-level analysis and sentence-level analysis attempt to classify the sentiment of a whole review to one of the predefined sentiment polarities, including positive, negative and sometimes neutral.
* While phrase-level analysis attempt to extract the sentiment polarity of each feature that a user expresses his/her attitude to the specific feature of a specific product.
* Zhang *et al.* propose a self-supervised and lexicon-based sentiment classification approach to determine sentiment polarity of a review that contains both textual words and emoticons. And they use sentiment for recommendation.
* Lee *et al.* propose a recommender system using the concept of Experts to find both novel and relevant recommendations. By analyzing the user ratings, they can recommend special experts to a target user based on the user population.

DISADVANTAGES OF EXISTING SYSTEM:

* The existing work mainly focuses on classifying users into binary sentiment (i.e. positive or negative), and they do not go further in mining user’s sentiment.
* The existing approaches mainly leverage product category information or tag information to study the interpersonal influence.

These methods are all restricted on the structured data, which is not always available on some websites. However, user reviews can provide us ideas in mining interpersonal inference and user preferences.

**1.3 Proposed System**

* We propose a sentiment-based rating prediction method in the framework of matrix factorization. In our work, we make use of social users’ sentiment to infer ratings.
* First, we extract product features from user reviews. Then, we find out the sentiment words, which are used to describe the product features. Besides, we leverage sentiment dictionaries to calculate sentiment of a specific user on an item/product.
* The main contributions of our approach are as follows:
* We propose a user sentimental measurement approach, which is based on the mined sentiment words and sentiment degree words from user reviews.
* We make use of sentiment for rating prediction. User sentiment similarity focuses on the user interest preferences. User sentiment influence reflects how the sentiment spreads among the trusted users. Item reputation similarity shows the potential relevance of items.
* We fuse the three factors: user sentiment similarity, interpersonal sentimental influence, and item reputation similarity into a probabilistic matrix factorization framework to carry out an accurate recommendation. The experimental results and discussions show that user's social sentiment that we mined is a key factor in improving rating prediction performances.

ADVANTAGES OF PROPOSED SYSTEM:

* We not only mine social user’s sentiment, but also explore interpersonal sentimental influence and item’s reputation. Finally, we take all of them into the recommender system.
* The purpose of our approach is to find effective clues from reviews and predict social users’ ratings.
* We fuse user sentiment similarity, inter personal sentiment influence, and item reputation similarity into a unified matrix factorization frame work to achieve the rating prediction task.

**2. LITERATURE REVIEW**

**1. Pipeline item-based collaborative filtering based on MapReduce**

**AUTHORS: Z. Zhao, C. Wang, Y. Wan, Z. Huang, J. Lai**

Its principle is based on the user's evaluation of items. The purpose is to find the similarity between users, and recommend items to the target user according to the records of the similar users. However, the number of customers and products keeps increasing at a high rate, which increases the cost to find out the recommendation list for each user.

**2. Review expert collaborative recommendation algorithm based on topic relationship**

**AUTHORS: S. Gao, Z. Yu, L. Shi, X. Yan, H.**

In this paper, we aim to determine review expert's rating by using the historical rating records and the final decision results on the previous projects, and by means of some rules, we construct a rating matrix for projects and experts.

**3. Item-based collaborative filtering recommendation algorithms**

**AUTHORS: B. Sarwar, G. Karypis, J. Konstan, and J. Reidl**

Recommender systems apply knowledge discovery techniques to the problem of making personalized recommendations for information, products or services during a live interaction.

**4. Scalable recommendation with social contextual information**

**AUTHORS: M. Jiang, P. Cui, F. Wang, W. Zhu, S. Yang**

In this paper, we investigate the social recommendation problem on the basis of psychology and sociology studies, which exhibit two important factors: individual preference and interpersonal influence.

**5. Semantic-based location recommendation with multimodal venue semantics**

**AUTHORS: X. Wang, Y. Zhao, L. Nie, Y. Gao**

In this paper, we aim to study the semantics of point-of-interest (POI) by exploiting the abundant heterogeneous user generated content (UGC) from different social networks. Our idea is to explore the text descriptions, photos, user check-in patterns, and venue context for location semantic similarity measurement.

**3. ANALYSIS**

**3.1 Scope**

* We have witnessed a flourish of review websites. It presents a great opportunity to share our viewpoints for various products we purchase.
* However, we face the information overloading problem.
* How to mine valuable information from reviews to understand a user’s preferences and make an accurate recommendation is crucial.
* Traditional recommender systems (RS) consider some factors, such as user’s purchase records, product category, and geographic location.
* In this work, we propose a sentiment-based rating prediction method (RPS) to improve prediction accuracy in recommender systems.

**3.2 Objectives**

* Firstly, we propose a social user sentimental measurement approach and calculate each user’s sentiment on items/products.
* Secondly, we not only consider a user’s own sentimental attributes but also take interpersonal sentimental influence into consideration.
* Then, we consider product reputation, which can be inferred by the sentimental distributions of a user set that reflect customers’ comprehensive evaluation.
* At last, we fuse three factors-user sentiment similarity, interpersonal sentimental influence, and item’s reputation similarity into our recommender system to make an accurate rating prediction.
* We conduct a performance evaluation of the three sentimental factors on a real-world dataset collected from Yelp.
* Our experimental results show the sentiment can well characterize user preferences, which help to improve the recommendation performance.

**3.3 Requirements**

**3.3.1 Functional Requirements**

**MODULES:**

* Data preprocessing for LDA
* Extracting product features
* User Sentimental Measurement
* Sentiment Evaluation

**Data preprocessing for LDA**

* We develop the data preprocessing for LDA. We have collected rating data set from http://www.yelp.com. We give this dataset as the input to our system. The data set are product items dataset, user ratings dataset and user feedback dataset. We have to separate dataset feedback and ratings based. The purpose of our approach is to find effective clues from reviews and predict social users’ ratings. In this module, we firstly extract product features from user review corpus, and then we introduce the method of identifying social users’ sentiment.
* The dataset are categories into three factors:

1. Item’s reputation

2. Interpersonal sentimental influence

3. User sentiment similarity.

**Extracting product features**

* We extract product features from textual reviews using LDA. We mainly want to get the product features including some named entities and some product/item/service attributes. LDA is a Bayesian model, which is utilized to model the relationship of reviews, topics and words
* To construct the vocabulary, we firstly regard each user’s review as a collection of words without considering the order. Then we filter out “Stop Words”, “Noise Words” and sentiment words, sentiment degree words, and negation words.
* A stop word can be identified as a word that has the same likelihood of occurring in those documents not relevant to a query as in those documents relevant to the query. For example, the “Stop Words” could be some prepositions, articles, and pronouns etc.. After words filtering, the input text is clear and without much interference for generating topics. All the unique words are constructed in the vocabulary 𝑉, each word has a label.
* From each topic, we have some frequent words. However, we need to filter the noisy features from the candidate set based on their co-occurrence with adjective words and their frequencies in background corpus.

**User Sentimental Measurement**

* We extend HowNet Sentiment Dictionary3 to calculate social user’s sentiment on items. In this module, we merge the positive sentiment words list and positive evaluation words list of HowNet Sentiment Dictionary into one list, and named it as POS-Words; also, we merge the negative sentiment words list and negative evaluation words list of HowNet Sentiment Dictionary into one list, and named it as NEG-Words.
* We develop five different levels in sentiment degree dictionary (**SDD**), which has 128 words in total. There are 52 words in the **Level-1,** which means the highest degree of sentiment, such as the words “*most*”, and “*best*”. And 48 words in the **Level-2**, which means higher degree of sentiment, such as the words “*better*”, and “*very*”. There are 12 words in the **Level-3**, such as the words “*more*”, and “*such*”. There are 9 words in the **Level-4,** such as the words “*a little*”, “*a bit*”, and “*more or less*”. And there are 7 words in the **Level-5,** such as the words “*less*”, “*bit*”, and “*not very*”. Also, we built the negation dictionary (**ND**) by collecting frequently-used negative prefix words, such as “*no*”, “*hardly*”, “*never*”, etc. These words are used to reverse the polarity of sentiment words.

**Sentiment Evaluation**

* We firstly divide the original review into several clauses by the punctuation mark. Then for each clause, we firstly look up the dictionary **SD** to find the sentiment words before the product features. A positive word is initially assigned with the score +1.0, while a negative word is assigned with the score −1.0. Secondly, we find out the sentiment degree words based on the dictionary **SDD** and take the sentiment degree words into consideration to strengthen sentiment for the found sentiment words. Finally, we check the negative prefix words based on the dictionary **ND** and add a negation check coefficient that has a default value of +1.0. If the sentiment word is preceded by an odd number of negative prefix words within the specified zone, we reverse the sentiment polarity, and the coefficient is set to −1.0.
* Each sentiment factor is described as follows: 1) User Sentiment Similarity, 2) Interpersonal Sentiment Influence, 3) Item Reputation Similarity
* We compare the performance of our method with the existing models on Yelp dataset. In the objective function of RPS, *k* is the dimension of user and item latent feature vectors.
* The experimental results show the high accuracy of RPS. Meanwhile, we demonstrate the importance of social friend factors (i.e. CircleCon2b, PRM) and explicit features (i.e. EFM) in a recommender system.

**3.3.2 Non-Functional Requirements**

**Usability:**

Prioritize the important functions of the system based on usage patterns. Frequently used functions should be tested for usability, as should complex and critical functions. Be sure to create a requirement for this.

**Reliability:**

Users have to trust the system, even after using it for a long time. Your goal should be a long MTBF (mean time between failures). Create a requirement that data created in the system will be retained for a number of years without the data being changed by the system. It’s a good idea to also include requirements that make it easier to monitor system performance.

**Performance:**

What should system response times be, as measured from any point, under what circumstances? Are there specific peak times when the load on the system will be unusually high? Think of stress periods, for example, at the end of the month or in conjunction with payroll disbursement.

**Supportability:**

The system needs to be **cost-effective to maintain**. Maintainability requirements may cover diverse levels of documentation, such as system documentation, as well as test documentation, e.g. which test cases and test plans will accompany the system.

**Performance requirements**   
Requirements about resources required, response time, transaction rates, throughput, benchmark specifications or anything else having to do with performance.

**Operating constraints**   
List any run-time constraints. This could include system resources, people, need software.

**Platform constraints**   
Discuss the target platform. Be as specific or general as the user requires. If the user doesn't care, there are still platform constraints.

**Accuracy and Precision**   
Requirements about the accuracy and precision of the data. (Do you know the difference?) Beware of 100% requirements; they often cost too much.

**Modifiability**   
Requirements about the effort required to make changes in the software. Often, the measurement is personnel effort (person- months).

**Portability**   
The effort required to move the software to a different target platform. The measurement is most commonly person-months or % of modules that need changing.

**Security**   
One or more requirements about protection of your system and its data. The measurement can be expressed in a variety of ways (effort, skill level, time) to break into the system.  Do not discuss solutions (e.g. passwords) in a requirements document.

**3.3.2.1 HARDWARE REQUIREMENTS:**

* System : Intel Core i5 4th gen
* Hard Disk : 500 GB.
* Monitor : LED display
* Mouse : Logitech.
* Ram : 4 GB.

**3.3.2.2 SOFTWARE REQUIREMENTS:**

* Operating system : Windows 7.
* Coding Language : JAVA/J2EE
* Data Base : MYSQL
* Tools : NetBeans

**4. DESIGN**

**UML DIAGRAMS**

UML stands for Unified Modeling Language. UML is a standardized general-purpose modeling language in the field of object-oriented software engineering. The standard is managed, and was created by, the Object Management Group.

The goal is for UML to become a common language for creating models of object oriented computer software. In its current form UML is comprised of two major components: a Meta-model and a notation. In the future, some form of method or process may also be added to; or associated with, UML.

The Unified Modeling Language is a standard language for specifying, Visualization, Constructing and documenting the artifacts of software system, as well as for business modeling and other non-software systems.

The UML represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems.

The UML is a very important part of developing objects oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects.

**GOALS**

The Primary goals in the design of the UML are as follows:

1. Provide users a ready-to-use, expressive visual modeling Language so that they can develop and exchange meaningful models.
2. Provide extendibility and specialization mechanisms to extend the core concepts.
3. Be independent of particular programming languages and development process.
4. Provide a formal basis for understanding the modeling language.
5. Encourage the growth of OO tools market.
6. Support higher level development concepts such as collaborations, frameworks, patterns and components.
7. Integrate best practices.

**USE CASE DIAGRAM:**

A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.

Split

Ratings

Fig 1:

**CLASS DIAGRAM:**

In software engineering, a class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among the classes. It explains which class contains information.

Fig 2:

Ratings Predication

Ratings based

Feedback based

Reputation()

Top most reputation()

User feedback ()

Level based ()

Product features()

Feedback report()

Suggest tags()

Rating Predication

Items based

Product based

us

Positive ratings ()

Negative ratings ()

Top most ratings ()

Interpersonal items()

ratings details()

**SEQUENCE DIAGRAM:**

A sequence diagram in Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams.

Database

Load dataset

Split product

Admin

Feedback read

Split items

Features based

Sentimental based

Items reputation

Top reputation items

Suggested items

Feedback analysis

Feedback Analysis based

Fig 3:

**ACTIVITY DIAGRAM:**

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modeling Language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control.

Fig 4:

Suggest by friend

Feedback based

Top reputation

Item reputation

Features

Analysis feedback

Levels

Sentimental

Positive

negative

predication

Ratings based

User Ratings

Start

splitting

Product based

Items based

**DATA FLOW DIAGRAM:**

1. The DFD is also called as bubble chart. It is a simple graphical formalism that can be used to represent a system in terms of input data to the system, various processing carried out on this data, and the output data is generated by this system.
2. The data flow diagram (DFD) is one of the most important modeling tools. It is used to model the system components. These components are the system process, the data used by the process, an external entity that interacts with the system and the information flows in the system.
3. DFD shows how the information moves through the system and how it is modified by a series of transformations. It is a graphical technique that depicts information flow and the transformations that are applied as data moves from input to output.
4. DFD is also known as bubble chart. A DFD may be used to represent a system at any level of abstraction. DFD may be partitioned into levels that represent increasing information flow and functional detail.

Fig 5:

Ratings based

Feedbacks

Top Most reputation

dataset

Split dataset

Items based

**User’s trusted**

Interpersonal count

Ratings based

All Reputation

Ratings

Suggested items

Feed backs

Reputation

interpersonal

Positive & Negative

**Item’s reputation**

**Sentiment**

Items

Level based

Product based

**SYSTEM DESIGN**

Systems design is the process of defining the architecture, modules, interfaces, and data for a system to satisfy specified requirements. Systems design could be seen as the application of systems theory to product development. There is some overlap with the disciplines of systems analysis, systems architecture and systems engineering.

### Logical design

The logical design of a system pertains to an abstract representation of the data flows, inputs and outputs of the system. This is often conducted via modelling, using an over-abstract (and sometimes graphical) model of the actual system. In the context of systems, designs are included. Logical design includes entity-relationship diagrams (ER diagrams).

### Physical design

The physical design relates to the actual input and output processes of the system. This is explained in terms of how data is input into a system, how it is verified/authenticated, how it is processed, and how it is displayed. In physical design, the following requirements about the system are decided.

**SYSTEM ARCHITECTURE:**

Fig 6:

****

**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.

**SOFTWARE ENVIRONMENT**

## Java Technology

Java technology is both a programming language and a platform.

### The Java Programming Language

### The Java programming language is a high-level language that can be characterized by all of the following buzzwords:

* + - Simple
    - Architecture neutral
    - Object oriented
    - Portable
    - Distributed
    - High performance
    - Interpreted
    - Multithreaded
    - Robust
    - Dynamic
    - Secure

With most programming languages, you either compile or interpret a program so that you can run it on your computer. The Java programming language is unusual in that a program is both compiled and interpreted. With the compiler, first you translate a program into an intermediate language called *Java byte codes* —the platform-independent codes interpreted by the interpreter on the Java platform. The interpreter parses and runs each Java byte code instruction on the computer. Compilation happens just once; interpretation occurs each time the program is executed. The following figure illustrates how this works.



Fig 7:

You can think of Java byte codes as the machine code instructions for the Java Virtual Machine (Java VM). Every Java interpreter, whether it’s a development tool or a Web browser that can run applets, is an implementation of the Java VM. Java byte codes help make “write once, run anywhere” possible. You can compile your program into byte codes on any platform that has a Java compiler. The byte codes can then be run on any implementation of the Java VM. That means that as long as a computer has a Java VM, the same program written in the Java programming language can run on Windows 2000, a Solaris workstation, or on an iMac.



### Fig 8:

### The Java Platform

A platform is the hardware or software environment in which a program runs. We’ve already mentioned some of the most popular platforms like Windows 2000, Linux, Solaris, and MacOS. Most platforms can be described as a combination of the operating system and hardware. The Java platform differs from most other platforms in that it’s a software-only platform that runs on top of other hardware-based platforms.

The Java platform has two components:

* The Java Virtual Machine (Java VM)
* The Java Application Programming Interface (Java API)

You’ve already been introduced to the Java VM. It’s the base for the Java platform and is ported onto various hardware-based platforms.

The Java API is a large collection of ready-made software components that provide many useful capabilities, such as graphical user interface (GUI) widgets. The Java API is grouped into libraries of related classes and interfaces; these libraries are known as packages. The next section, What Can Java Technology Do? Highlights what functionality some of the packages in the Java API provide.

The following figure depicts a program that’s running on the Java platform. As the figure shows, the Java API and the virtual machine insulate the program from the hardware.

Fig 9: 

Native code is code that after you compile it, the compiled code runs on a specific hardware platform. As a platform-independent environment, the Java platform can be a bit slower than native code. However, smart compilers, well-tuned interpreters, and just-in-time byte code compilers can bring performance close to that of native code without threatening portability.

## What Can Java Technology Do?

The most common types of programs written in the Java programming language are applets and applications*.* If you’ve surfed the Web, you’re probably already familiar with applets. An applet is a program that adheres to certain conventions that allow it to run within a Java-enabled browser.

However, the Java programming language is not just for writing cute, entertaining applets for the Web. The general-purpose, high-level Java programming language is also a powerful software platform. Using the generous API, you can write many types of programs.

An application is a standalone program that runs directly on the Java platform. A special kind of application known as a server serves and supports clients on a network. Examples of servers are Web servers, proxy servers, mail servers, and print servers. Another specialized program is a servlet. A servlet can almost be thought of as an applet that runs on the server side. Java Servlets are a popular choice for building interactive web applications, replacing the use of CGI scripts. Servlets are similar to applets in that they are runtime extensions of applications. Instead of working in browsers, though, servlets run within Java Web servers, configuring or tailoring the server.

How does the API support all these kinds of programs? It does so with packages of software components that provides a wide range of functionality. Every full implementation of the Java platform gives you the following features:

* **The essentials**: Objects, strings, threads, numbers, input and output, data structures, system properties, date and time, and so on.
* **Applets**: The set of conventions used by applets.
* **Networking**: URLs, TCP (Transmission Control Protocol), UDP (User Data gram Protocol) sockets, and IP (Internet Protocol) addresses.
* **Internationalization**: Help for writing programs that can be localized for users worldwide. Programs can automatically adapt to specific locales and be displayed in the appropriate language.
* **Security**: Both low level and high level, including electronic signatures, public and private key management, access control, and certificates.
* **Software components**: Known as JavaBeans, can plug into existing component architectures.
* **Object serialization**: Allows lightweight persistence and communication via Remote Method Invocation (RMI).
* **Java Database Connectivity (JDBCTM)**: Provides uniform access to a wide range of relational databases.

The Java platform also has APIs for 2D and 3D graphics, accessibility, servers, collaboration, telephony, speech, animation, and more. The following figure depicts what is included in the Java 2 SDK.



## Fig 10:

## How Will Java Technology Change My Life?

We can’t promise you fame, fortune, or even a job if you learn the Java programming language. Still, it is likely to make your programs better and requires less effort than other languages. We believe that Java technology will help you do the following:

* **Get started quickly**: Although the Java programming language is a powerful object-oriented language, it’s easy to learn, especially for programmers already familiar with C or C++.
* **Write less code**: Comparisons of program metrics (class counts, method counts, and so on) suggest that a program written in the Java programming language can be four times smaller than the same program in C++.
* **Write better code**: The Java programming language encourages good coding practices, and its garbage collection helps you avoid memory leaks. Its object orientation, its JavaBeans component architecture, and its wide-ranging, easily extendible API let you reuse other people’s tested code and introduce fewer bugs.
* **Develop programs more quickly**: Your development time may be as much as twice as fast versus writing the same program in C++. Why? You write fewer lines of code and it is a simpler programming language than C++.
* **Avoid platform dependencies with 100% Pure Java**: You can keep your program portable by avoiding the use of libraries written in other languages. The 100% Pure JavaProduct Certification Program has a repository of historical process manuals, white papers, brochures, and similar materials online.
* **Write once, run anywhere**: Because 100% Pure Java programs are compiled into machine-independent byte codes, they run consistently on any Java platform.
* **Distribute software more easily**: You can upgrade applets easily from a central server. Applets take advantage of the feature of allowing new classes to be loaded “on the fly,” without recompiling the entire program.

### ODBC

Microsoft Open Database Connectivity (ODBC) is a standard programming interface for application developers and database systems providers. Before ODBC became a de facto standard for Windows programs to interface with database systems, programmers had to use proprietary languages for each database they wanted to connect to. Now, ODBC has made the choice of the database system almost irrelevant from a coding perspective, which is as it should be. Application developers have much more important things to worry about than the syntax that is needed to port their program from one database to another when business needs suddenly change.

Through the ODBC Administrator in Control Panel, you can specify the particular database that is associated with a data source that an ODBC application program is written to use. Think of an ODBC data source as a door with a name on it. Each door will lead you to a particular database. For example, the data source named Sales Figures might be a SQL Server database, whereas the Accounts Payable data source could refer to an Access database. The physical database referred to by a data source can reside anywhere on the LAN.

The ODBC system files are not installed on your system by Windows 95. Rather, they are installed when you setup a separate database application, such as SQL Server Client or Visual Basic 4.0. When the ODBC icon is installed in Control Panel, it uses a file called ODBCINST.DLL. It is also possible to administer your ODBC data sources through a stand-alone program called ODBCADM.EXE. There is a 16-bit and a 32-bit version of this program and each maintains a separate list of ODBC data sources.

From a programming perspective, the beauty of ODBC is that the application can be written to use the same set of function calls to interface with any data source, regardless of the database vendor. The source code of the application doesn’t change whether it talks to Oracle or SQL Server. We only mention these two as an example. There are ODBC drivers available for several dozen popular database systems. Even Excel spreadsheets and plain text files can be turned into data sources. The operating system uses the Registry information written by ODBC Administrator to determine which low-level ODBC drivers are needed to talk to the data source (such as the interface to Oracle or SQL Server). The loading of the ODBC drivers is transparent to the ODBC application program. In a client/server environment, the ODBC API even handles many of the network issues for the application programmer.

The advantages of this scheme are so numerous that you are probably thinking there must be some catch. The only disadvantage of ODBC is that it isn’t as efficient as talking directly to the native database interface. ODBC has had many detractors make the charge that it is too slow. Microsoft has always claimed that the critical factor in performance is the quality of the driver software that is used. In our humble opinion, this is true. The availability of good ODBC drivers has improved a great deal recently. And anyway, the criticism about performance is somewhat analogous to those who said that compilers would never match the speed of pure assembly language. Maybe not, but the compiler (or ODBC) gives you the opportunity to write cleaner programs, which means you finish sooner. Meanwhile, computers get faster every year.

**JDBC**

In an effort to set an independent database standard API for Java; Sun Microsystems developed Java Database Connectivity, or JDBC. JDBC offers a generic SQL database access mechanism that provides a consistent interface to a variety of RDBMSs. This consistent interface is achieved through the use of “plug-in” database connectivity modules, or drivers. If a database vendor wishes to have JDBC support, he or she must provide the driver for each platform that the database and Java run on.

To gain a wider acceptance of JDBC, Sun based JDBC’s framework on ODBC. As you discovered earlier in this chapter, ODBC has widespread support on a variety of platforms. Basing JDBC on ODBC will allow vendors to bring JDBC drivers to market much faster than developing a completely new connectivity solution.

JDBC was announced in March of 1996. It was released for a 90 day public review that ended June 8, 1996. Because of user input, the final JDBC v1.0 specification was released soon after.

The remainder of this section will cover enough information about JDBC for you to know what it is about and how to use it effectively. This is by no means a complete overview of JDBC. That would fill an entire book.

### JDBC Goals

Few software packages are designed without goals in mind. JDBC is one that, because of its many goals, drove the development of the API. These goals, in conjunction with early reviewer feedback, have finalized the JDBC class library into a solid framework for building database applications in Java.

The goals that were set for JDBC are important. They will give you some insight as to why certain classes and functionalities behave the way they do. The eight design goals for JDBC are as follows:

1. SQL Level API

The designers felt that their main goal was to define a SQL interface for Java. Although not the lowest database interface level possible, it is at a low enough level for higher-level tools and APIs to be created. Conversely, it is at a high enough level for application programmers to use it confidently. Attaining this goal allows for future tool vendors to “generate” JDBC code and to hide many of JDBC’s complexities from the end user.

1. SQL Conformance

SQL syntax varies as you move from database vendor to database vendor. In an effort to support a wide variety of vendors, JDBC will allow any query statement to be passed through it to the underlying database driver. This allows the connectivity module to handle non-standard functionality in a manner that is suitable for its users.

1. JDBC must be implemental on top of common database interfaces   
    The JDBC SQL API must “sit” on top of other common SQL level APIs. This goal allows JDBC to use existing ODBC level drivers by the use of a software interface. This interface would translate JDBC calls to ODBC and vice versa.
2. Provide a Java interface that is consistent with the rest of the Java system

Because of Java’s acceptance in the user community thus far, the designers feel that they should not stray from the current design of the core Java system.

1. Keep it simple

This goal probably appears in all software design goal listings. JDBC is no exception. Sun felt that the design of JDBC should be very simple, allowing for only one method of completing a task per mechanism. Allowing duplicate functionality only serves to confuse the users of the API.

1. Use strong, static typing wherever possible

Strong typing allows for more error checking to be done at compile time; also, less error appear at runtime.

1. Keep the common cases simple

Because more often than not, the usual SQL calls used by the programmer are simple SELECT’s, INSERT’s, DELETE’s and UPDATE’s, these queries should be simple to perform with JDBC. However, more complex SQL statements should also be possible.

Finally we decided to proceed the implementation using Java Networking.

And for dynamically updating the cache table we go for MS Access database.

Java has two things: a programming language and a platform.

Java is a high-level programming language that is all of the following

Simple Architecture-neutral

Object-oriented Portable

Distributed High-performance

Interpreted multithreaded

Robust Dynamic

Secure

Java is also unusual in that each Java program is both compiled and interpreted. With a compile you translate a Java program into an intermediate language called Java byte codes the platform-independent code instruction is passed and run on the computer.

Compilation happens just once; interpretation occurs each time the program is executed. The figure illustrates how this works.

**Java Program**

**Compilers**

**Interpreter**

**My Program**

Fig 11:

You can think of Java byte codes as the machine code instructions for the Java Virtual Machine (Java VM). Every Java interpreter, whether it’s a Java development tool or a Web browser that can run Java applets, is an implementation of the Java VM. The Java VM can also be implemented in hardware.

Java byte codes help make “write once, run anywhere” possible. You can compile your Java program into byte codes on my platform that has a Java compiler. The byte codes can then be run any implementation of the Java VM. For example, the same Java program can run Windows NT, Solaris, and Macintosh.

**Networking**

**TCP/IP stack**

The TCP/IP stack is shorter than the OSI one:

Fig 12:

TCP is a connection-oriented protocol; UDP (User Datagram Protocol) is a connectionless protocol.

### IP datagram’s

The IP layer provides a connectionless and unreliable delivery system. It considers each datagram independently of the others. Any association between datagram must be supplied by the higher layers. The IP layer supplies a checksum that includes its own header. The header includes the source and destination addresses. The IP layer handles routing through an Internet. It is also responsible for breaking up large datagram into smaller ones for transmission and reassembling them at the other end.

### UDP

UDP is also connectionless and unreliable. What it adds to IP is a checksum for the contents of the datagram and port numbers. These are used to give a client/server model - see later.

### TCP

TCP supplies logic to give a reliable connection-oriented protocol above IP. It provides a virtual circuit that two processes can use to communicate.

### Internet addresses

In order to use a service, you must be able to find it. The Internet uses an address scheme for machines so that they can be located. The address is a 32 bit integer which gives the IP address. This encodes a network ID and more addressing. The network ID falls into various classes according to the size of the network address.

### Network address

Class A uses 8 bits for the network address with 24 bits left over for other addressing. Class B uses 16 bit network addressing. Class C uses 24 bit network addressing and class D uses all 32.

### Subnet address

Internally, the UNIX network is divided into sub networks. Building 11 is currently on one sub network and uses 10-bit addressing, allowing 1024 different hosts.

### Host address

8 bits are finally used for host addresses within our subnet. This places a limit of 256 machines that can be on the subnet.

### 

### Total address

Fig 13:

The 32 bit address is usually written as 4 integers separated by dots.

### Port addresses

A service exists on a host, and is identified by its port. This is a 16 bit number. To send a message to a server, you send it to the port for that service of the host that it is running on. This is not location transparency! Certain of these ports are "well known".

### Sockets

A socket is a data structure maintained by the system to handle network connections. A socket is created using the call socket. It returns an integer that is like a file descriptor. In fact, under Windows, this handle can be used with Read File and Write File functions.

#include <sys/types.h>

#include <sys/socket.h>

int socket(int family, int type, int protocol);

Here "family" will be AF\_INET for IP communications, protocol will be zero, and type will depend on whether TCP or UDP is used. Two processes wishing to communicate over a network create a socket each. These are similar to two ends of a pipe - but the actual pipe does not yet exist.

**JFree Chart**

JFreeChart is a free 100% Java chart library that makes it easy for developers to display professional quality charts in their applications. JFreeChart's extensive feature set includes:

A consistent and well-documented API, supporting a wide range of chart types;

A flexible design that is easy to extend, and targets both server-side and client-side applications;

Support for many output types, including Swing components, image files (including PNG and JPEG), and vector graphics file formats (including PDF, EPS and SVG);

JFreeChart is "open source" or, more specifically, free software. It is distributed under the terms of the GNU Lesser General Public Licence (LGPL), which permits use in proprietary applications.

## 1. Map Visualizations

Charts showing values that relate to geographical areas. Some examples include: (a) population density in each state of the United States, (b) income per capita for each country in Europe, (c) life expectancy in each country of the world. The tasks in this project include:

Sourcing freely redistributable vector outlines for the countries of the world, states/provinces in particular countries (USA in particular, but also other areas);

Creating an appropriate dataset interface (plus default implementation), a rendered, and integrating this with the existing XYPlot class in JFreeChart;

Testing, documenting, testing some more, documenting some more.

**2. Time Series Chart Interactivity**

Implement a new (to JFreeChart) feature for interactive time series charts --- to display a separate control that shows a small version of ALL the time series data, with a sliding "view" rectangle that allows you to select the subset of the time series data to display in the main chart.

**3. Dashboards**

There is currently a lot of interest in dashboard displays. Create a flexible dashboard mechanism that supports a subset of JFreeChart chart types (dials, pies, thermometers, bars, and lines/time series) that can be delivered easily via both Java Web Start and an applet.

**4. Property Editors**

The property editor mechanism in JFreeChart only handles a small subset of the properties that can be set for charts. Extend (or reimplement) this mechanism to provide greater end-user control over the appearance of the charts.

**J2ME (Java 2 Micro edition):-**

Sun Microsystems defines J2ME as "a highly optimized Java run-time environment targeting a wide range of consumer products, including pagers, cellular phones, screen-phones, digital set-top boxes and car navigation systems." Announced in June 1999 at the JavaOne Developer Conference, J2ME brings the cross-platform functionality of the Java language to smaller devices, allowing mobile wireless devices to share applications. With J2ME, Sun has adapted the Java platform for consumer products that incorporate or are based on small computing devices.

**1. General J2ME architecture**

Fig 14: 

J2ME uses configurations and profiles to customize the Java Runtime Environment (JRE). As a complete JRE, J2ME is comprised of a configuration, which determines the JVM used, and a profile, which defines the application by adding domain-specific classes. The configuration defines the basic run-time environment as a set of core classes and a specific JVM that run on specific types of devices. We'll discuss configurations in detail in the the profile defines the application; specifically, it adds domain-specific classes to the J2ME configuration to define certain uses for devices. We'll cover profiles in depth in the following graphic depicts the relationship between the different virtual machines, configurations, and profiles. It also draws a parallel with the J2SE API and its Java virtual machine. While the J2SE virtual machine is generally referred to as a JVM, the J2ME virtual machines, KVM and CVM, are subsets of JVM. Both KVM and CVM can be thought of as a kind of Java virtual machine -- it's just that they are shrunken versions of the J2SE JVM and are specific to J2ME.

**2. Developing J2ME applications**

Introduction In this section, we will go over some considerations you need to keep in mind when developing applications for smaller devices. We'll take a look at the way the compiler is invoked when using J2SE to compile J2ME applications. Finally, we'll explore packaging and deployment and the role pre verification plays in this process.

**3. Design considerations for small devices**

Developing applications for small devices requires you to keep certain strategies in mind during the design phase. It is best to strategically design an application for a small device before you begin coding. Correcting the code because you failed to consider all of the "gotchas" before developing the application can be a painful process. Here are some design strategies to consider:

\* Keep it simple. Remove unnecessary features, possibly making those features a separate, secondary application.

\* Smaller is better. This consideration should be a "no brainer" for all developers. Smaller applications use less memory on the device and require shorter installation times. Consider packaging your Java applications as compressed Java Archive (jar) files.

\* Minimize run-time memory use. To minimize the amount of memory used at run time, use scalar types in place of object types. Also, do not depend on the garbage collector. You should manage the memory efficiently yourself by setting object references to null when you are finished with them. Another way to reduce run-time memory is to use lazy instantiation, only allocating objects on an as-needed basis. Other ways of reducing overall and peak memory use on small devices are to release resources quickly, reuse objects, and avoid exceptions.

**4. Configurations overview**

The configuration defines the basic run-time environment as a set of core classes and a specific JVM that run on specific types of devices. Currently, two configurations exist for J2ME, though others may be defined in the future:

\* Connected Limited Device Configuration (CLDC) is used specifically with the KVM for 16-bit or 32-bit devices with limited amounts of memory. This is the configuration (and the virtual machine) used for developing small J2ME applications. Its size limitations make CLDC more interesting and challenging (from a development point of view) than CDC. CLDC is also the configuration that we will use for developing our drawing tool application. An example of a small wireless device running small applications is a Palm hand-held computer.

\* Connected Device Configuration (CDC) is used with the C virtual machine (CVM) and is used for 32-bit architectures requiring more than 2 MB of memory. An example of such a device is a Net TV box.

**5. J2ME profiles**

What is a J2ME profile?

As we mentioned earlier in this tutorial, a profile defines the type of device supported. The Mobile Information Device Profile (MIDP), for example, defines classes for cellular phones. It adds domain-specific classes to the J2ME configuration to define uses for similar devices. Two profiles have been defined for J2ME and are built upon CLDC: KJava and MIDP. Both KJava and MIDP are associated with CLDC and smaller devices. Profiles are built on top of configurations. Because profiles are specific to the size of the device (amount of memory) on which an application runs, certain profiles are associated with certain configurations.

A skeleton profile upon which you can create your own profile, the Foundation Profile, is available for CDC.

**Profile 1: KJava**

KJava is Sun's proprietary profile and contains the KJava API. The KJava profile is built on top of the CLDC configuration. The KJava virtual machine, KVM, accepts the same byte codes and class file format as the classic J2SE virtual machine. KJava contains a Sun-specific API that runs on the Palm OS. The KJava API has a great deal in common with the J2SE Abstract Windowing Toolkit (AWT). However, because it is not a standard J2ME package, its main package is com.sun.kjava. We'll learn more about the KJava API later in this tutorial when we develop some sample applications.

**Profile 2: MIDP**

MIDP is geared toward mobile devices such as cellular phones and pagers. The MIDP, like KJava, is built upon CLDC and provides a standard run-time environment that allows new applications and services to be deployed dynamically on end user devices. MIDP is a common, industry-standard profile for mobile devices that is not dependent on a specific vendor. It is a complete and supported foundation for mobile application development. MIDP contains the following packages, the first three of which are core CLDC packages, plus three MIDP-specific packages.

\* java.lang

\* java.io

\* java.util

\* javax.microedition.io

\* javax.microedition.lcdui

\* javax.microedition.midlet

\* javax.microedition.rms

**5. Sample Code**

//allproductitems.java

package rating.prediction.based;

import java.sql.Connection;

import java.sql.ResultSet;

import java.sql.ResultSetMetaData;

import java.sql.SQLException;

import java.sql.Statement;

import java.util.Vector;

import java.util.logging.Level;

import java.util.logging.Logger;

import javax.swing.JOptionPane;

import javax.swing.table.DefaultTableModel;

public class allproductitems extends javax.swing.JFrame {

Connection con;

static Statement st;

static ResultSet rs,rs1,rs2,rs3,rs4,rs5;

static String path;

static int row;

static String s,name;

String filename;

private int i;

//Creates new form allproductitems

public allproductitems() {

try {

initComponents();

con = dobc.getConne();

st = (Statement) con.createStatement();

} catch (SQLException ex) {

Logger.getLogger(allproductitems.class.getName()).log(Level.SEVERE, null, ex);

}

}

@SuppressWarnings("unchecked")

// <editor-fold defaultstate="collapsed" desc="Generated Code">

private void initComponents() {

jScrollPane2 = new javax.swing.JScrollPane();

jTable1 = new javax.swing.JTable();

jButton1 = new javax.swing.JButton();

jTextField1 = new javax.swing.JTextField();

jLabel1 = new javax.swing.JLabel();

jButton2 = new javax.swing.JButton();

jLabel2 = new javax.swing.JLabel();

jLabel3 = new javax.swing.JLabel();

setDefaultCloseOperation(javax.swing.WindowConstants.EXIT\_ON\_CLOSE);

getContentPane().setLayout(new org.netbeans.lib.awtextra.AbsoluteLayout());

jTable1.setFont(new java.awt.Font("Baskerville Old Face", 1, 14)); // NOI18N

jTable1.setModel(new javax.swing.table.DefaultTableModel(

new Object [][] {

},

new String [] {

}

));

jScrollPane2.setViewportView(jTable1);

getContentPane().add(jScrollPane2, new org.netbeans.lib.awtextra.AbsoluteConstraints(50, 70, 340, 230));

jButton1.setBackground(new java.awt.Color(153, 153, 0));

jButton1.setFont(new java.awt.Font("Times New Roman", 1, 11)); // NOI18N

jButton1.setText("VIEW\_ALL\_PRODUCT");

jButton1.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(java.awt.event.ActionEvent evt) {

jButton1ActionPerformed(evt);

}

});

getContentPane().add(jButton1, new org.netbeans.lib.awtextra.AbsoluteConstraints(410, 150, 160, 30));

jTextField1.setText("jTextField1");

getContentPane().add(jTextField1, new org.netbeans.lib.awtextra.AbsoluteConstraints(290, 330, 160, 30));

jLabel1.setFont(new java.awt.Font("Times New Roman", 1, 14)); // NOI18N

jLabel1.setText("OVER\_ALL\_ITEMS\_COUNT");

getContentPane().add(jLabel1, new org.netbeans.lib.awtextra.AbsoluteConstraints(70, 330, 180, 30));

jButton2.setBackground(new java.awt.Color(153, 153, 0));

jButton2.setFont(new java.awt.Font("Times New Roman", 1, 11)); // NOI18N

jButton2.setText("NEXT");

jButton2.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(java.awt.event.ActionEvent evt) {

jButton2ActionPerformed(evt);

}

});

getContentPane().add(jButton2, new org.netbeans.lib.awtextra.AbsoluteConstraints(440, 210, 90, 30));

jLabel2.setFont(new java.awt.Font("Bodoni MT Black", 1, 18)); // NOI18N

jLabel2.setText("ALL PRODUCT ITEMS");

getContentPane().add(jLabel2, new org.netbeans.lib.awtextra.AbsoluteConstraints(90, 30, 260, -1));

jLabel3.setIcon(new javax.swing.ImageIcon(getClass().getResource("/rating/prediction/based/bgg.jpg"))); // NOI18N

jLabel3.setText("jLabel3");

getContentPane().add(jLabel3, new org.netbeans.lib.awtextra.AbsoluteConstraints(0, 0, 610, 390));

java.awt.Dimension screenSize = java.awt.Toolkit.getDefaultToolkit().getScreenSize();

setBounds((screenSize.width-628)/2, (screenSize.height-433)/2, 628, 433);

}

// </editor-fold>

private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {

try {

// TODO add your handling code here:

Vector col = new Vector();

Vector data = new Vector();

rs = (ResultSet) st.executeQuery("SELECT \* FROM c\_items\_product ");

ResultSetMetaData md = (ResultSetMetaData) rs.getMetaData();

int columns = md.getColumnCount();

int count=0;

for (int i = 1; i <= columns; i++) {

col.addElement(md.getColumnName(i));

}

while (rs.next()) {

count++;

Vector row = new Vector(columns);

for (int i = 1; i <= columns; i++) {

row.addElement(rs.getObject(i));

}

data.addElement(row);

}

jTextField1.setText(Integer.toString(count));

con.close();

DefaultTableModel model = new DefaultTableModel(data, col);

jTable1.setModel(model);

JOptionPane.showMessageDialog(null, "over all product items");

} catch (SQLException ex) {

Logger.getLogger(allproductitems.class.getName()).log(Level.SEVERE, null, ex);

}

}

private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {

// TODO add your handling code here:

new alluserratings().setVisible(true);

}

// @param args the command line arguments

public static void main(String args[]) {

/\* Set the Nimbus look and feel \*/

//<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional) ">

/\* If Nimbus (introduced in Java SE 6) is not available, stay with the default look and feel.

\* For details see http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html

\*/

try {

for (javax.swing.UIManager.LookAndFeelInfo info : javax.swing.UIManager.getInstalledLookAndFeels()) {

if ("Nimbus".equals(info.getName())) {

javax.swing.UIManager.setLookAndFeel(info.getClassName());

break;

}

}

} catch (ClassNotFoundException ex) {

java.util.logging.Logger.getLogger(allproductitems.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);

} catch (InstantiationException ex) {

java.util.logging.Logger.getLogger(allproductitems.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);

} catch (IllegalAccessException ex) {

java.util.logging.Logger.getLogger(allproductitems.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);

} catch (javax.swing.UnsupportedLookAndFeelException ex) {

java.util.logging.Logger.getLogger(allproductitems.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);

}

//</editor-fold>

/\* Create and display the form \*/

java.awt.EventQueue.invokeLater(new Runnable() {

public void run() {

new allproductitems().setVisible(true);

}

});

}

// Variables declaration - do not modify

private javax.swing.JButton jButton1;

private javax.swing.JButton jButton2;

private javax.swing.JLabel jLabel1;

private javax.swing.JLabel jLabel2;

private javax.swing.JLabel jLabel3;

private javax.swing.JScrollPane jScrollPane2;

private javax.swing.JTable jTable1;

private javax.swing.JTextField jTextField1;

// End of variables declaration

}

**6. TESTING**

**SYSTEM TESTING**

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, subassemblies, 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.

**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.

**Test Cases:**

|  |  |  |  |
| --- | --- | --- | --- |
| Test Id | Test Cases | Fig No. | Output |
| 1 | Unit Testing | Fig15, Fig16 | This testing is done in different modules and has successfully executed. |
| 2 | Integration Testing | Fig15&Fig16, Fig16&Fig17 | This testing is done between the two components where all are been executed successfully. |
| 3 | White Box Testing | Fig15-Fig20 | This testing is done between all the software modules and the java codes where it has been executed successfully. |
| 4 | Acceptance Testing | Fig 20 | This testing has been done on the graph module of the project where the output is been executed successfully. |

**7. RESULTS**

**Output**

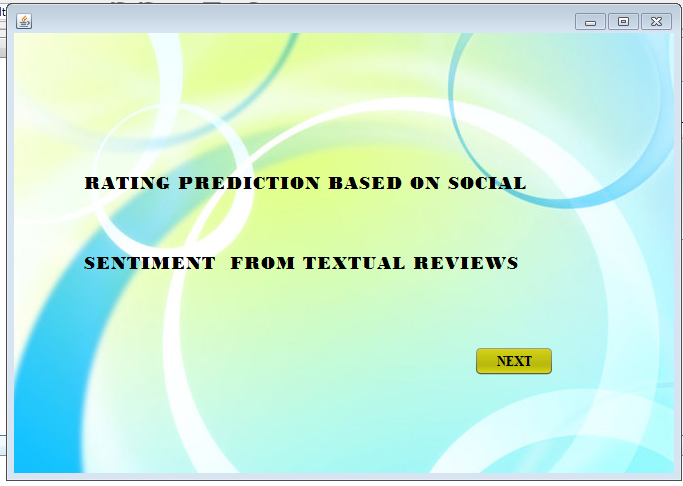
Fig 15: 

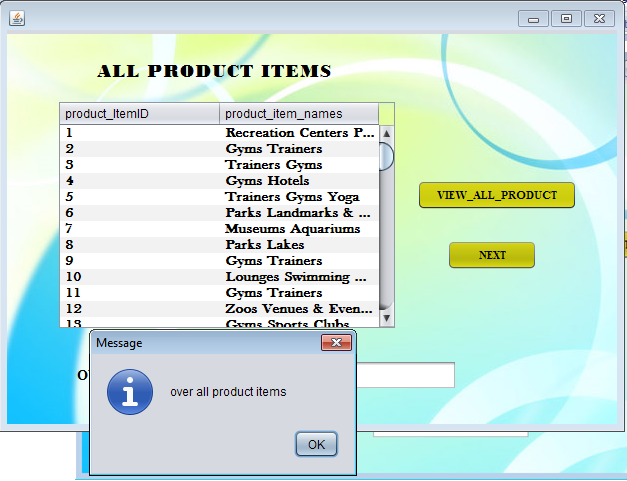
Fig 16: 

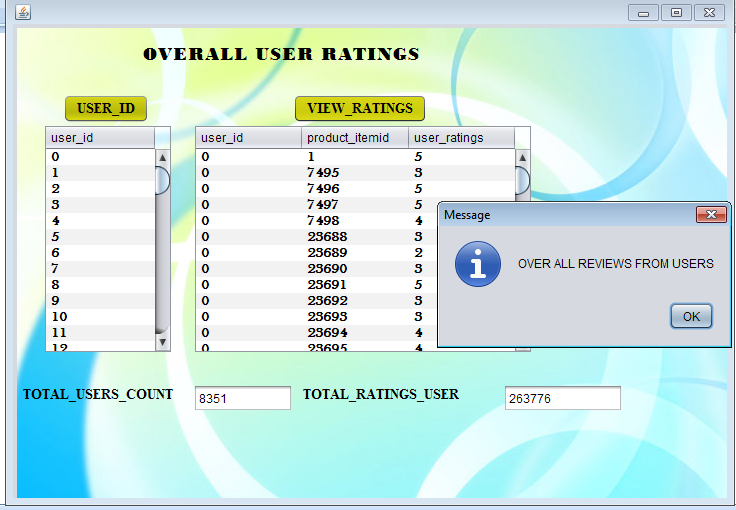
Fig 17: 

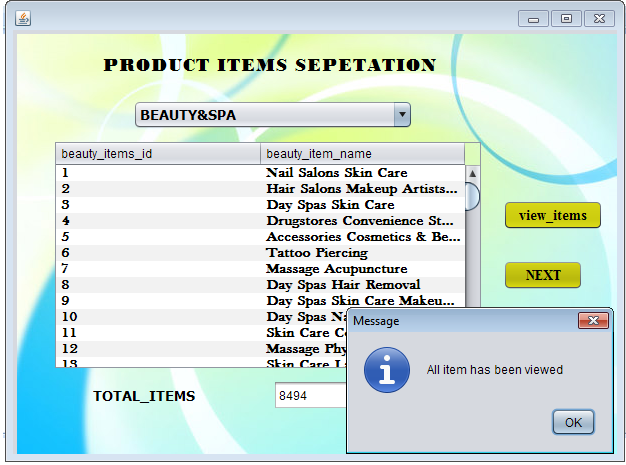
Fig 18: 

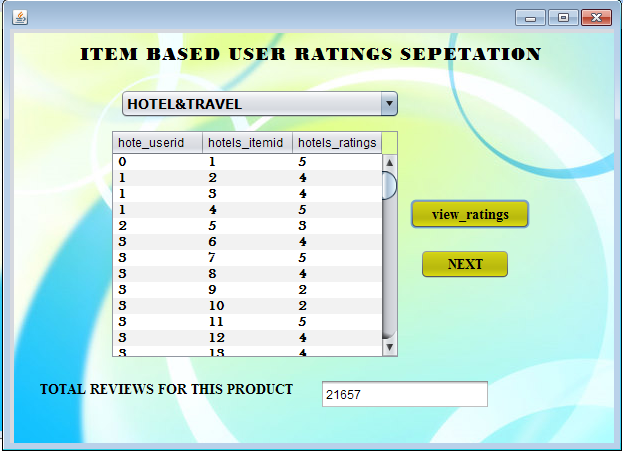
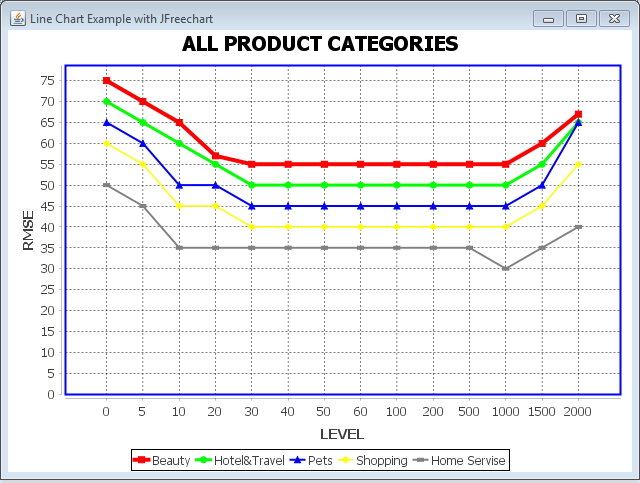
Fig 19: 

Fig 20: 

**8. Conclusion and Future work**

A recommendation model is proposed by mining sentiment information from social user’s reviews. We use user sentiment similarity, interpersonal sentiment influence, and item reputation similarity into a unified matrix factorization frame work to achieve the rating prediction task. In particular, we use social users’ sentiment to denote user preferences. Besides, we build a new relationship named interpersonal sentiment influence between the user and friends, which reflect show users’ friends influence users in a sentimental angle. What is more, as long as we obtain user’s textual reviews, we can quantitivley measure user’s sentiment, and we leverage items’ sentiment distribution among users to infer item’s reputation. The experiment results demonstrate that the three sentimental factors make great contributions to the rating prediction. Also, it shows significant improvements over existing approaches on a real-world dataset. In our future work, we can consider more linguistic rules when analyzing the context, and we can enrich the sentiment dictionaries to apply fine-grained sentiment analysis. Besides, we can adapt or develop other hybrid factorization models such as tensor factorization or deep learning technique to integrate phrase-level sentiment analysis.

**9. REFERENCES**

[1] Xiaojiang Lei, Xueming Qian, Member, IEEE, and Guoshuai Zhao “Rating Prediction Based on Social Sentiment From Textual Reviews”

[2] R. Salakhutdinov and A. Mnih, “Probabilistic matrix factorization,” in Proc. NIPS, 2007, pp. 1257–1264.

[3] D. M. Blei, A.Y. Ng, and M. I. Jordan, “Latent Dirichlet allocation,” J.Mach. Learn. Res. vol. 3, pp. 993–1022, 2003.

[4] M. Jamali and M. Ester, “A matrix factorization technique with trust propagation for recommendation in social networks,” in Proc. ACM Conf. Recommender Syst., 2010, pp. 135–142.

[5] T. Kawashima, T. Ogawa, and M. Haseyama, “A rating prediction method for e-commerce application using ordinal regression based on LDA with multi-modal features,” in Proc. IEEE 2nd Global Conf. Consum. Electron, 2013, pp. 260–261.

[6] G. Ganu, N. Elhadad, and A Marian, “Beyond the stars: Improving rating predictions using Review text content,” in Proc. 12th Int. Workshop Web Databases, 2009, pp. 1–6.

[7] J. Xu, X. Zheng, and W. Ding, “Personalized recommendation based on reviews and ratings alleviating the sparsity problem of collaborative filtering,” in Proc. IEEE 9th Int. Conf. e-Bus. Eng., Sep. 2012, pp.9–16.

[8] X. Qian, H. Feng, G. Zhao, and T. Mei, “Personalized recommendation combining user interest and social circle,” IEEE Trans. Knowl. Data Eng., vol. 26, no. 7, pp. 1763–1777, Jul. 2014.

[9] Y. Ren, J. Shen, J. Wang, J. Han, and S. Lee, “Mutual verifiable provable data auditing in public cloud storage,” J. Internet Technol., vol. 16, no. 2, pp. 317–323, 2015.

[10] M. Jiang, P. Cui, F. Wang, W. Zhu, and S. Yang, “Scalable recommendation with social contextual information,” IEEE Trans. Knowl. Data Eng., vol. 26, no. 11, pp. 2789–2802, Nov. 2014.

[11] V. Suresh, A. Veilumuthu, and A. Krishnamurthy, “A non-syntactic approach for text sentiment classification with stop words,” in Proc. 20th Int. Conf. Companion World Wide Web, Mar.–Apr. 2011, pp. 137–138.