A Thesis/Project/Dissertation Report

on

RECOMMENDATION SYSTEM

Submitted in partial fulfillment of the

requirement for the award of the degreeof

B.Tech in Computer Science Engineering

(Established under Galgotias University Uttar Pradesh Act No. 14 of 2011)

Under The Supervision of
Dr.Dileep Kumar Yadav
Professor

Submitted By

Akul Joshi
18021120004/18SCSE1120005

SCHOOL OF COMPUTING SCIENCE AND ENGINEERING
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING /
/ DEPARTMENT OF COMPUTERAPPLICATION
GALGOTIAS UNIVERSITY, GREATER NOIDA
INDIA
December, 2021

SCHOOL OF COMPUTING SCIENCE AND ENGINEERING
GALGOTIAS UNIVERSITY, GREATER NOIDA

CANDIDATE'S DECLARATION

I/We hereby certify that the work which is being presented in the thesis/project/dissertation, entitled **"RECOMMENDATION SYSTEM "** in partial fulfillment of the requirements for the award of the B.Tech in Computer Science Engineering –submitted in the School of Computing Science and Engineering of Galgotias University, Greater Noida, is an original work carried out during the period of September, 2021 to December and 2021, under the supervision of Dr.Dileep Kumar Yadav (Professor), Department of Computer Science and Engineering/Computer Application and Information and Science, of School of Computing Science and Engineering , Galgotias University, Greater Noida

The matter presented in the thesis/project/dissertation has not been submitted by me/us for the award of any other degree of this or any other places.

Akul Joshi 18SCSE1120005

This is to certify that the above statement made by the candidates is correct to the best of my knowledge.

CERTIFICATE

The Final Thesis/Project/ Dissertation Viva-Voce examination of Akul Joshi 18SCSE1120005 has been held on - ___________________and his/her work is recommended for the award of B.Tech. Computer Science Engineering.

Signature of Examiner(s) **Signature of Supervisor(s)**

Signature of Project Coordinator **Signature of Dean**

Date: November, 2013

Place: Greater Noida

Acknowledgement

I feel great pleasure in submitting this project paper on Recommendation system. I wish to express true sense of gratitude towards my project guide, Dr.Dileep Kumar Yadav sir who at every stage in the study of this project, contributed his valuable guidance and helped to solve every and every problem. Our great obligation would remain due towards Dr.Dileep Kumar Yadav Sir who was a constant inspiration during my project. He provided with an opportunity to undertake the project at Galgotias University College of Engineering, Greater Noida. I feel highly indebted to them who provided us with all our project requirements, and did much beyond my expectations to bring out the best in me.

Abstract

Recommendation systems have expanded our options online whether streaming a movie online on Netflix or buying clothes from Myntra. This Movie recommendation system's goal is to assist movie buffs by recommending what film to watch without requiring them to go through the time-consuming and complex process of selecting from a vast number of films. The goal of this project is to reduce human effort by proposing movies based on the user's preferences. Along with movie recommendations the model also show the movie posters. I developed an Engine that includes Data Science techniques such as Machine Learning, Text sentiment analysis, and Deep Learning to assist users. The project can have several unique features, such as suggesting similar movies from a movie poster using reverse image search and performing sentimental analysis on movie reviews. To sum up, I have concentrated on developing a system that minimizes human effort.

Contents

List of Figures

CHAPTER-1

Introduction

A recommendation system is a type of information filtering system which attempts to predict the preferences of a user, and make suggests based on these preferences. There are a wide variety of applications for recommendation systems. These have become increasingly popular over the last few years and are now utilized in most online platforms that we use. The content of such platforms varies from movies, music, books and videos, to friends and stories on social media platforms, to products on e-commerce websites, to people on professional and dating websites, to search results returned on Google. Often, these systems are able to collect information about a users choices, and can use this information to improve their suggestions in the future. For example, Facebook can monitor your interaction with various stories on your feed in order to learn what types of stories appeal to you. Sometimes, the recommender systems can make improvements based on the activities of a large number of people. For example, if Amazon observes that a large number of customers who buy the latest Apple Macbook also buy a USB-C-toUSB Adapter, they can recommend the Adapter to a new user who has just added a Macbook to his cart. Due to the advances in recommender systems, users constantly expect good recommendations. They have a low threshold for services that are not able to make appropriate suggestions. If a music streaming app is not able to predict and play music that the user likes, then the user will simply stop using it. This has led to a high emphasis by tech companies on improving their recommendation systems. However, the problem is more complex than it seems. Every user has different preferences and likes. In addition, even the taste of a single user can vary depending on a large number of factors, such as mood, season, or type of activity the user is doing. For example, the type of music one would like to hear while exercising differs greatly from the type of music he'd listen to when cooking dinner. Another issue that recommendation systems have to solve is the exploration vs exploitation problem. They must explore new domains to discover more about the user, while still making the most of what is already known about of the user. Three main approaches are used for our recommender systems. One is Demographic Filtering i.e They offer generalized recommendations to every user, based on movie popularity and/or

7
genre. The System recommends the same movies to users with similar demographic features.

Since each user is different , this approach is considered to be too simple. The basic idea behind this system is that movies that are more popular and critically acclaimed will have a higher probability of being liked by the average audience. Second is content-based filtering, where we try to profile the users interests using information collected, and recommend items based on that profile. The other is collaborative filtering, where we try to group similar users together and use information about the group to make recommendations to the user

Machine learning (ML) is a type of artificial intelligence (AI) that allows software applications to become more accurate at predicting outcomes without being explicitly programmed to do so. Machine learning algorithms use historical data as input to predict new output values.
Machine learning is important because it gives enterprises a view of trends in customer behavior and business operational patterns, as well as supports the development of new products. Many of today's leading companies, such as Facebook, Google and Uber, make machine learning a central part of their operations. Machine learning has become a significant competitive differentiator for many companies.
Supervised learning is type of machine learning, data scientists supply algorithms with labeled training data and define the variables they want the algorithm to assess for correlations. Both the input and the output of the algorithm is specified.

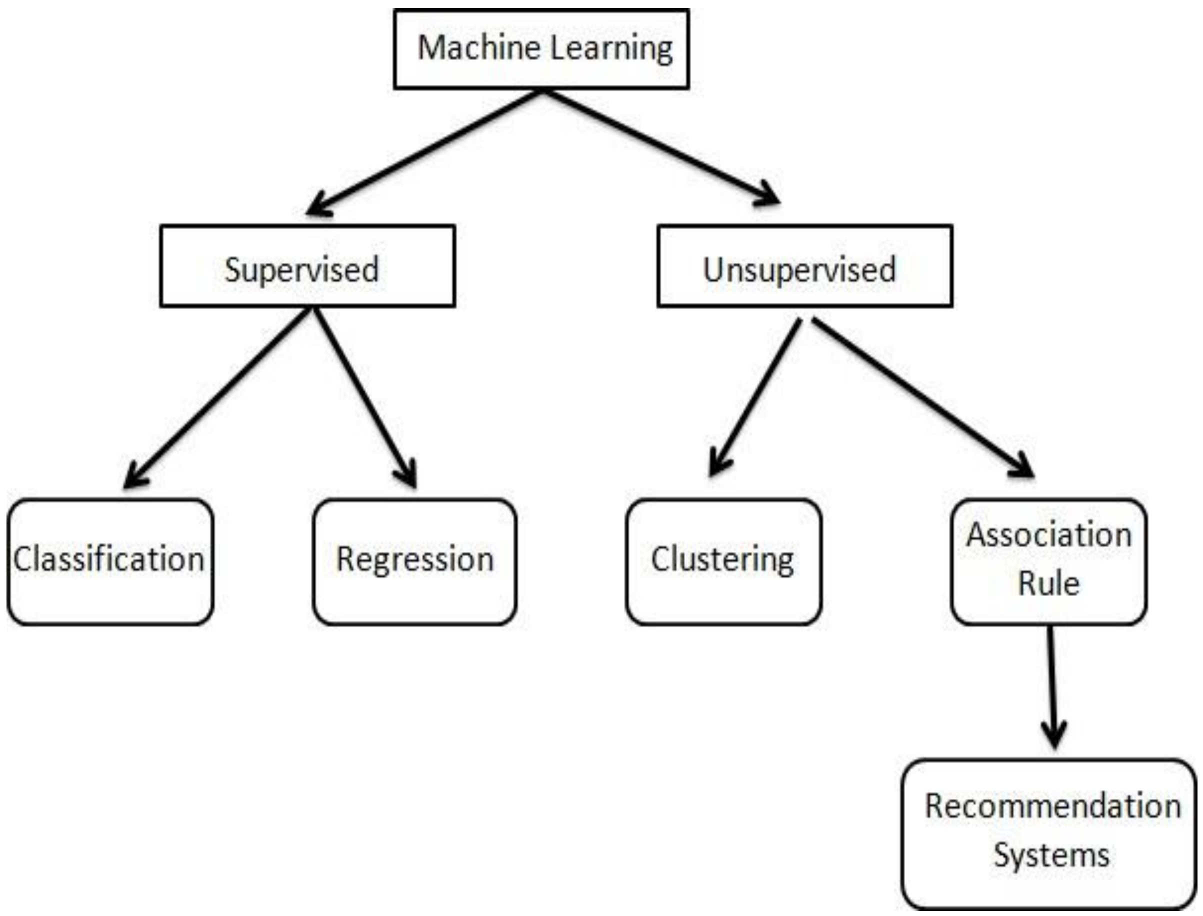


Fig. Machine Learning diagram

Supervised machine learning requires the data scientist to train the algorithm with both labeled inputs and desired outputs. Supervised learning algorithms are good for the following tasks:

1. Binary classification: Dividing data into two categories.
2. Multi-class classification: Choosing between more than two types of answers.
3. Regression modeling: Predicting continuous values.
4. Assembling: Combining the predictions of multiple machine learning models to produce an accurate prediction.

➢ 1.1 Formulation of problem

For building a recommender system from scratch, we face several different problems. Currently there are a lot of recommender systems based on the user information, so what should we do if the website has not gotten enough users. After that, we will solve the representation of a movie, which is how a system can understand a movie. That is the precondition for comparing similarity between two movies. Movie features such as genre, actor and director is a way that can categorize movies. But for each feature of the movie, there should be different weight for them and each of them plays a different role for recommendation. So we get these questions:

- Which dataset to use.
- What kind of movie features can be used for the recommender system.
- How to calculate the similarity between two movies.
- How to showcase movie posters along with movie names.

A. SOLUTION:
In order to achieve a project goal, the first step is to create an adequate domain read, so the book study will be conducted. The whole project is supported with a large amount of movie data to select the quantitative research method. In philosophical speculation, positivism is preferred because the project is experimental and character check. The research method is to deduct money as the development of our research will be evaluated by extracting and evaluating theory. Ex post facto research is our research strategy, movie data has already been collected we do not change the independent variables. We use experiments to compile a movie data. Computer statistics are used to analyze data because the result is based in algorithm development. For quality assurance, we have a detailed description of a test authentication algorithm. The same results will be made if we use the same data many times, reliably. We guarantee the same data leading to the same result by different researcher

➢ 1.2 Tools and technologies used

1.2.1 Hardware Requirements

RAM : At least 256 MB of RAM. The amount of RAM needed depends on the number of concurrent client connections, and whether the server and multiplexor are deployed on the same host.
Disk Space: Approximately 300 MB required for Instant Messaging Server software. **Processor**: Minimun 1.9 gigahertz (GHz) x86- or x64-bit dual core processor with SSE2 instruction set and recommended 3.3 gigahertz (GHz) or faster 64-bit dual core processor with SSE2 instruction set.
Memory: Minimum 2-GB RAM and recommended 4-GB RAM or more

1.2.2 Software Requirements

Python: is an interpreted high-level general-purpose programming language. Its design philosophy emphasizes code readability with its use of significant indentation. Its language constructs as well as its object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects.

Jupyter Notebook: The Jupyter Notebook is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations and narrative text. Uses include: data cleaning and transformation, numerical simulation, statistical modeling, data visualization, machine learning, and much more.

Heroku: Heroku is an ecosystem of cloud services, which can be used to instantly extend applications with fully-managed services. Using an existing, high-quality service is something that empowers developers - they can build more, faster, by using trusted services that provide the functionality that they require

CHAPTER-2

Literature Survey/Project Design

Over the past decade, a large number of recommendation systems for a variety of domains have been developed and are in use. These recommendation systems use a variety of methods such as content based approach, collaborative approach, knowledge based approach, utility based approach, hybrid approach, etc. Most of the online recommendation systems for a variety of items use ratings from previous users to make recommendations to current users with similar interests. One such system was designed by Jung, Harris, Webster and Herlocker (2004) for improving search results. The system encourages users to enter longer and more informative search queries, and collects ratings from users as to whether search results meet their information need or not. These ratings are then used to make recommendations to later users with similar needs

Fig. literature rurvey

➢ 2.1 Use Case Diagram

- Use case diagrams model the functionality of a system using actors and use cases. Use cases are a set of actions, services, and functions that the system needs to perform.

- First function is to give input i.e. set of movies with their details like movie name, language and to which genre it belongs.

- Apriori is an algorithm for frequent item set mining and association rule learning over relational databases. It proceeds by identifying the frequent individual items in the database and extending them to larger and larger item sets as long as those item sets appear sufficiently often in the database.

- Then recommends the movie based on Content-based Recommendation and Collaborative Filtering Algorithm.

- Based on the suggestion given by the system user will choose the movie.

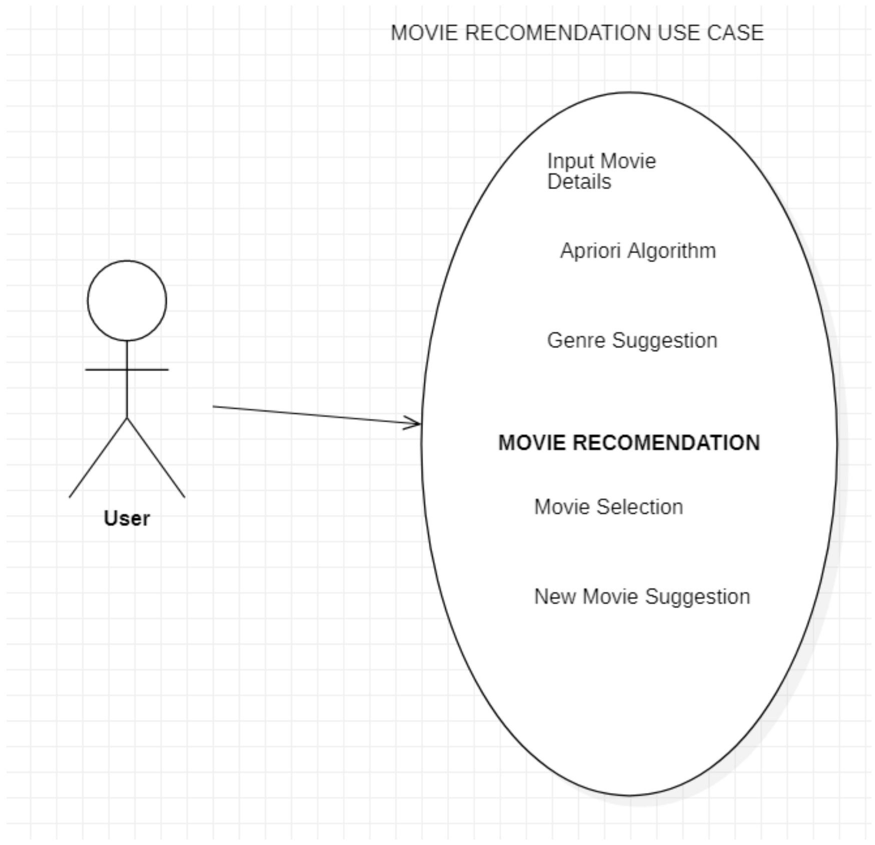


Fig. Use Case

Chapter-3
Methodology

In order to achieve the goal of the project, the first process is to do enough back- ground study, so the literature study will be conducted. The whole project is based on a big amount of movie data so that we choose quantitative research method. For philosophical assumption, positivism is selected because the project is experi- mental and testing character. The research approach is deductive approach as the improvement of our research will be tested by deducing and testing a theory. Ex post facto research is our research strategy, the movie data is already collected and we don’t change the independent variables. We use experiments to collect movie data. Computational mathematics is used data analysis because the result is based on improvement of algorithm. For the quality assurance, we have a detail explana- tion of algorithm to ensure test validity. The similar results will be generated when we run the same data multiple times, which is for reliability. We ensure the same data leading to same result by different researchers.

Chapter- 4
Approach

There are various types of recommender systems with different approaches and some of them are classified as below:

1. Demographic Filtering- They offer generalized recommendations to every user, based on movie popularity and/or genre. The System recommends the same movies to users with similar demographic features. Since each user is different , this approach is considered to be too simple. The basic idea behind this system is that movies that are more popular and critically acclaimed will have a higher probability of being liked by the average audience.

Before getting started with this -

- We need a metric to score or rate movie
- Calculate the score for every movie
- Sort the scores and recommend the best rated movie to the users.

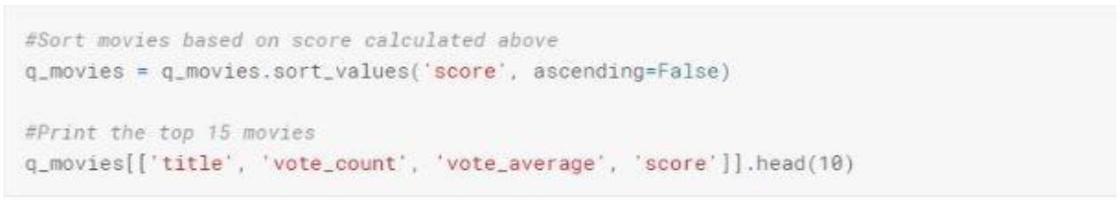
We can use the average ratings of the movie as the score but using this won't be fair enough since a movie with 8.9 average rating and only 3 votes cannot be considered better than the movie with 7.8 as as average rating but 40 votes. So, I'll be using IMDB's weighted rating

(wr) which is given as :-

$$\text{Weighted Rating (WR)} = \left(\frac{v}{v+m} . R\right) + \left(\frac{m}{v+m} . C\right)$$

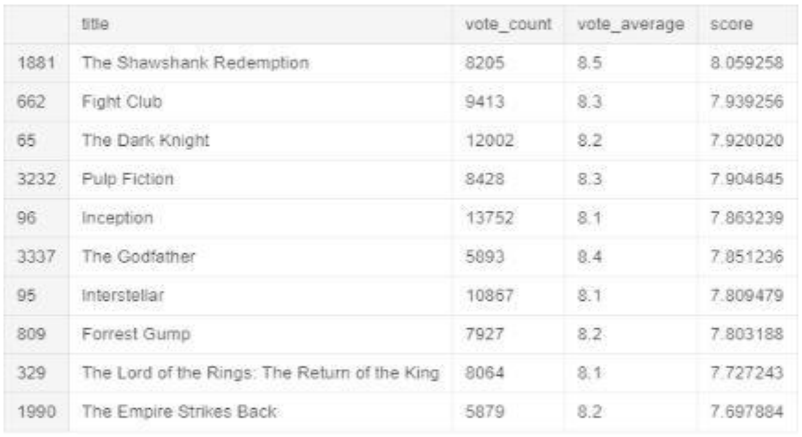
where,

- v is the number of votes for the movie;
- m is the minimum votes required to be listed in the chart;
- R is the average rating of the movie; And
- C is the mean vote across the whole report



```
#Sort movies based on score calculated above
q_movies = q_movies.sort_values('score', ascending=False)

#Print the top 15 movies
q_movies[['title', 'vote_count', 'vote_average', 'score']].head(10)
```



	title	vote_count	vote_average	score
1881	The Shawshank Redemption	8205	8.5	8.059258
662	Fight Club	9413	8.3	7.939256
65	The Dark Knight	12002	8.2	7.920020
3232	Pulp Fiction	8428	8.3	7.904645
96	Inception	13752	8.1	7.863239
3337	The Godfather	5893	8.4	7.851236
95	Interstellar	10867	8.1	7.809479
809	Forrest Gump	7927	8.2	7.803188
329	The Lord of the Rings: The Return of the King	8064	8.1	7.727243
1990	The Empire Strikes Back	5879	8.2	7.697884

2. Content-based Filtering Systems: In content-based filtering, items are recommended based on comparisons between item profile and user profile. A user profile is content that is found to be relevant to the user in form of keywords(or features). A user profile might be seen as a set of assigned keywords (terms, features) collected by algorithm from items found relevant (or interesting) by the user. A set of keywords (or features) of an item is the Item profile. For example, consider a scenario in which a person goes to buy his favorite cake 'X' to a pastry. Unfortunately, cake 'X' has been sold out and as a result of

this the shopkeeper recommends the person to buy cake ‘Y’ which is made up of ingredients similar to cake ‘X’. This is an instance of content-based filtering.

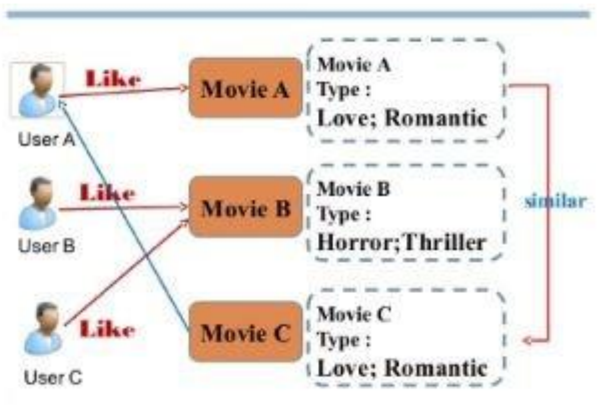


Fig. Content Based Filtering

We will be using the cosine similarity to calculate a numeric quantity that denotes the similarity between two movies. We use the cosine similarity score since it is independent of magnitude and is relatively easy and fast to calculate. Mathematically, it is defined as follows:

$$\text{similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\| \|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \sqrt{\sum_{i=1}^{n} B_i^2}},$$

We are now in a good position to define our recommendation function. These are the following steps we'll follow :-

- Get the index of the movie given its title.
- Get the list of cosine similarity scores for that particular movie with all movies. Convert it into a list of tuples where the first element is its position and the second is the

similarity score.

- Sort the aforementioned list of tuples based on the similarity scores; that is, the second element.
- Get the top 10 elements of this list. Ignore the first element as it refers to self (the movie most similar to a particular movie is the movie itself).

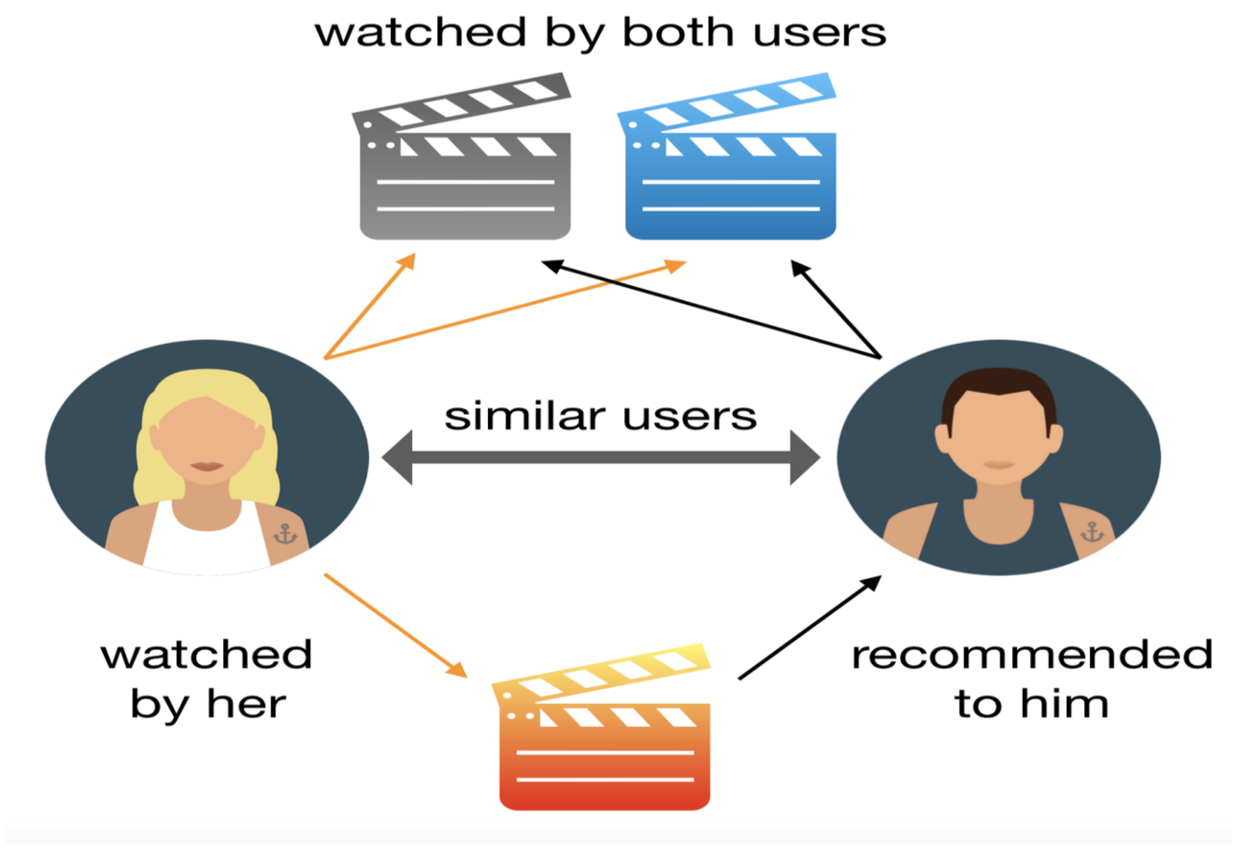


Fig. Content Based recommendation

- Return the titles corresponding to the indices of the top elements.

While our system has done a decent job of finding movies with similar plot descriptions, the quality of recommendations is not that great. "The Dark Knight Rises" returns all Batman movies while it is more likely that the people who liked that movie are more inclined to enjoy other Christopher Nolan movies. This is something that cannot be captured by the present system.

Advantages of content-based filtering are:

- They capable of recommending unrated items.
- We can easily explain the working of recommender system by listing the Content features of an item.
- Content-based recommender systems use need only the rating of the concerned user,and not any other user of the system.

Disadvantages of content-based filtering are:

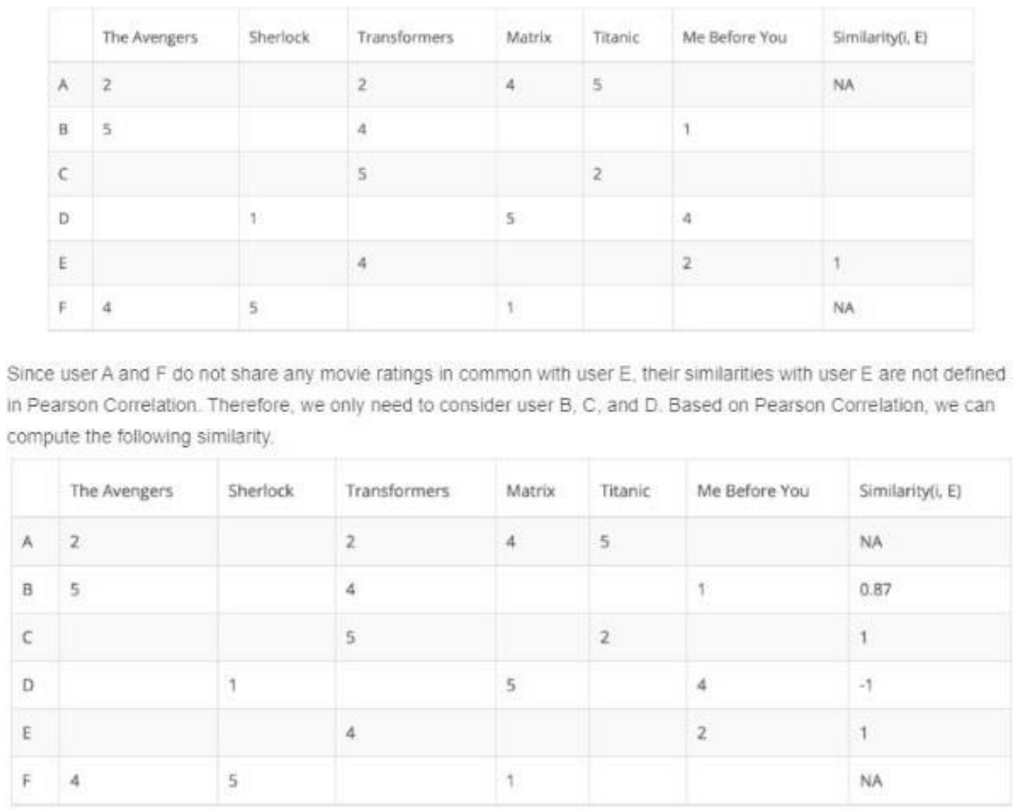
- It does not work for a new user who has not rated any item yet as enough ratings are required contentbased recommender evaluates the user preferences and provides accurate recommendations.
- No recommendation of serendipitous items.
- Limited Content Analysis- The recommend does not work if the system fails to distinguish the items hat a user likes from the items that he does not like.

3. Collaborative filtering based systems: Our content based engine suffers from some severe limitations. It is only capable of suggesting movies which are close to a certain movie. That is, it is not capable of capturing tastes and providing recommendations across genres.

Also, the engine that we built is not really personal in that it doesn't capture the personal tastes and biases of a user. Anyone querying our engine for recommendations based on a movie will receive the same recommendations for that movie, regardless of who she/he is.

Therefore, in this section, we will use a technique called Collaborative Filtering to make recommendations to Movie Watchers. It is basically of two types:-

a) <u>User based filtering</u>- These systems recommend products to a user that similar users have liked. For measuring the similarity between two users we can either use pearson correlation or cosine similarity. This filtering technique can be illustrated with an example. In the following matrix's, each row represents a user, while the columns correspond to different movies except the last one which records the similarity between that user and the target user. Each cell represents the rating that the user gives to that movie. Assume user E is the target.



	The Avengers	Sherlock	Transformers	Matrix	Titanic	Me Before You	Similarity(i, E)
A	2		2	4	5		NA
B	5		4			1	
C			5		2		
D		1		5		4	
E			4			2	1
F	4	5		1			NA

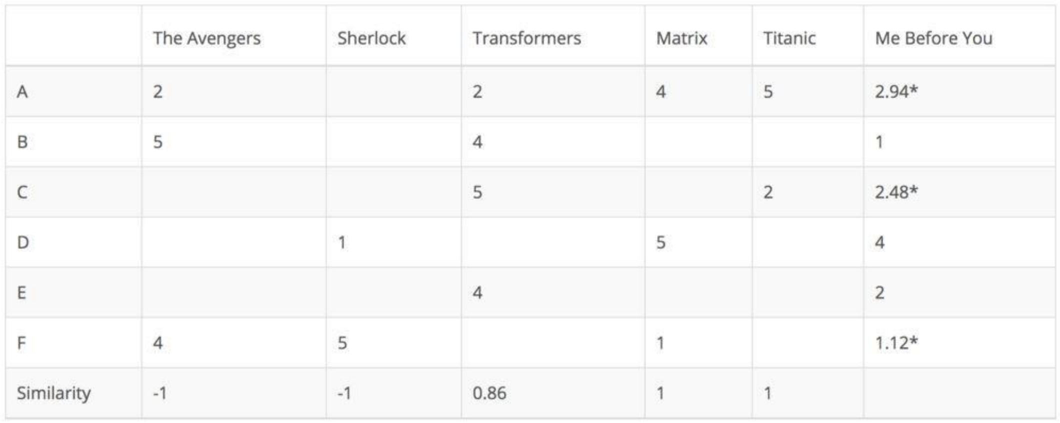
Since user A and F do not share any movie ratings in common with user E, their similarities with user E are not defined in Pearson Correlation. Therefore, we only need to consider user B, C, and D. Based on Pearson Correlation, we can compute the following similarity.

	The Avengers	Sherlock	Transformers	Matrix	Titanic	Me Before You	Similarity(i, E)
A	2		2	4	5		NA
B	5		4			1	0.87
C			5		2		1
D		1		5		4	-1
E			4			2	1
F	4	5		1			NA

Fig. User Based Filtering

Although computing user-based CF is very simple, it suffers from several problems. One main issue is that users' preference can change over time. It indicates that precomputing the matrix based on their neighboring users may lead to bad performance. To tackle this problem, we can apply item-based CF.

b) <u>Item Based Collaborative Filtering</u> - Instead of measuring the similarity between users, the item-based CF recommends items based on their similarity with the items that the target user rated. Likewise, the similarity can be computed with Pearson Correlation or Cosine Similarity. The major difference is that, with item-based collaborative filtering, we fill in the blank vertically, as oppose to the horizontal manner that user-based CF does. The following table shows how to do so for the movie Me Before.



	The Avengers	Sherlock	Transformers	Matrix	Titanic	Me Before You
A	2		2	4	5	2.94*
B	5		4			1
C			5		2	2.48*
D		1		5		4
E			4			2
F	4	5		1		1.12*
Similarity	-1	-1	0.86	1	1	

Fig. Item Based Filtering

It successfully avoids the problem posed by dynamic user preference as item-based CF is more static. However, several problems remain for this method. First, the main issue is *scalability*. The computation grows with both the customer and the product. The worst case complexity is O(mn) with m users and n items. In addition, *sparsity* is another concern. Take a look at the above table again. Although there is only one user that rated both Matrix and Titanic rated, the similarity between them is 1. In extreme cases, we can have millions of users and the similarity between two fairly different movies could be very high simply because they have similar rank for the only user who ranked them both.

➢ 4.1 Techniques for User Modeling and Item Analysis

In a formal definition, recommendation systems address the following problem: Given a set of items A, a subset B of A, with $|B| << |A|$ and the values f(b) $\forall b \in B$ of a function f(x), A-> {0,1} , where the closed type of f(x) is unknown, find an estimate of f(x), namely g(x), A-> {0,1}, and a subset C of A, with $|C| << |A|$ and $B \cap C = \emptyset$, so that the probability P(g(c) = f(c) = 1), $\forall c \in C$, is maximized. The common target set of f(x) and g(x) can be also numerical, using a type of threshold to distinguish between different classes. For the rest of this paper, we will consider the target set to be boolean, unless noted differently. Or is it? This definition can be interpreted as such: A is the set of items being offered to the user through a web service, B is the subset of items already rated negatively (0) or positively (1) and C is the subset of items to be recommended. The function f(x) represents the like or dislike of any item by the user, which obviously is not known and the function g(x) is our estimate for the user's interests, derived from a variety of methods, examined later in this section. The size of sets B and C need to be very much smaller (denoted by the use of <<) than A for practical reasons , such as the rating of a reasonable number of items by the user used as training data or recommending a number of items that the user can examine in a short time and can be presented in the limited visual space of a web page. Our goal is to create the best estimate possible g(x) of the user's interest function f(x), for every item in set A, in order to recommend items that have a high probability to match the user's interests. In the case of content-based recommendation systems, we use the item's description and the user's rating of a small set of items in order to create an estimate of the user's interest. Then, we use this estimate to decide which items fit best the user's interests and therefore should be recommended. The problem is actually a problem of classification or regression , with both categories been studied extensively and can be solved with a variety of algorithms derived from the field of machine learning. The problem can be broken down into 3 separate sub-problems, which address different areas of the problem. First of all , before we can apply mathematical models and algorithms to create an estimate of the user's interest, we must create a representation of the real world items in a way that they can be processed by an algorithm. In other words, we must decide the description that will be used to classify the items. Then, we must decide what model or algorithm we will use to create an estimate of the user's interests and implement it into a software system. Finally, we use this estimate in order to decide if we should recommend an item or not. The items that seem to match the user's interests the most are selected and displayed to the user. In the rest of this section , we will analyze every subproblem and present a variety of known solutions to them, along with the limitations , advantages and disadvantages of those solutions. Furthermore , we display how this subproblems are linked to solve the original problem and how decisions in one of those subproblems might affect the decisions for the other ones. Moreover, when available, we provide experimental data on the accuracy of the results produced by the method.

➢ 4.2 Decision Trees

Decision trees are a methodology that derives from the field of Decision Theory. Given a training set, they create a model that relies on the values of items in certain characteristics in order to accurately classify a previously unclassified item, using the least number of characteristics possible. Every node of the tree represents a set of items, with restricted values on their characteristics. The root node does not follow that rule, including the whole set of items, whereas the leaf nodes consist of items that belong only to one class or have restricted values in every characteristic. To create those sets, data metrics from the field of Information Theory are used, commonly Expected Information Gain and Information Entropy, in order to select the characteristic that partitions the set of items the most , depending on the values the items have in this particular characteristic. This partitioning of one set into several subsets is represented by branches in the decision tree, leading to different nodes, each for every value of the selected characteristic. While this technique is suitable for discrete value characteristics, continuous value ones need to be subjected to a process of discretization. An example of a decision tree follows, using the characteristics of atmospheric conditions during a tennis game, namely outlook, humidity and wind, in order to determine whether or not to play on a certain day. A decision tree for play

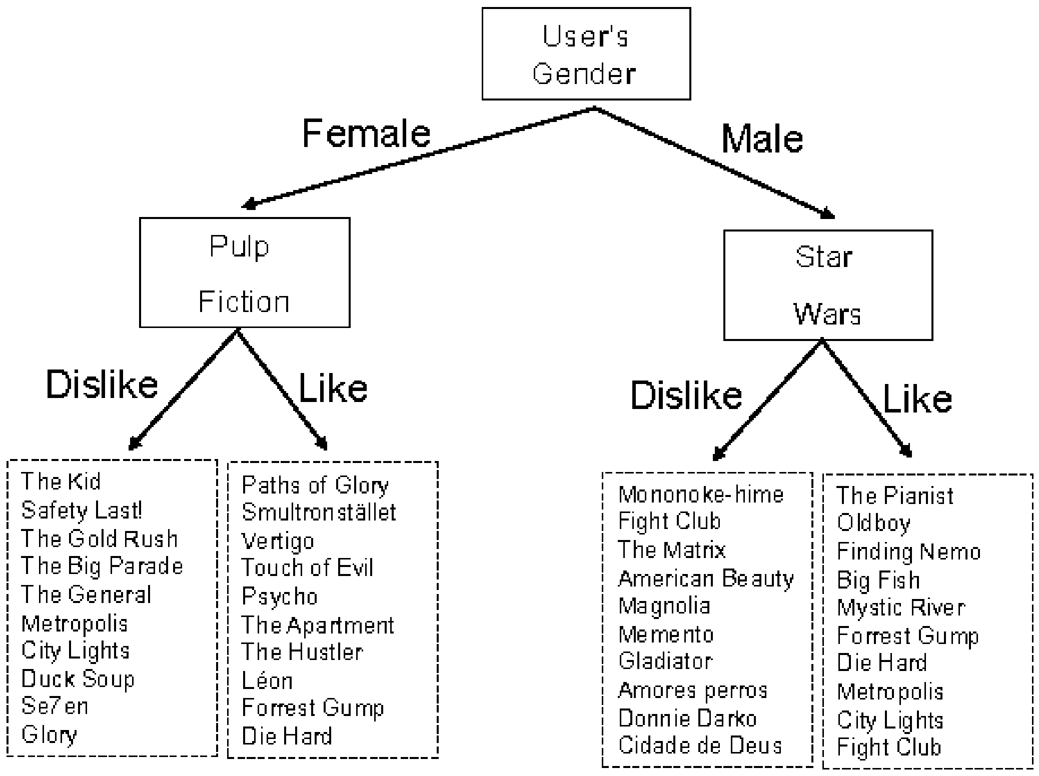


Fig Decision Trees

➢ 4.3 Examples of Content Based Recommendation Systems

Nowadays, recommendation systems have been widely adopted in the Web. Although some users may have concerns about the use of their private data in order to receive a greater quality of service, most users are amazed by the services that recommender systems offer them and trust the company's confidence agreement. Companies that operate in the web services market, have acknowledged this fact and have adopted accordingly their policies. Especially in the e-commerce market, recommender systems are viewed as one of the most valuable assets a company possesses. This can be attributed to the unique opportunity that recommender systems offer, that allows a company to provide personalized services to every customer, with minimum overhead and cost, in an automated way.

Content-based recommendation systems were the first approach to recommender systems, being developed since the mid 90's and they were quickly adopted by major web companies on their web sites. Therefore, although many of the recommender systems employed in the Web are content-based, they are not dominating the field. In the early years of recommender systems, the main competition was collaborative filtering systems, however the hybrid version prevailed, combining the advantages of both methodologies, thus offering better results. However, many companies still prefer the usage of content-based systems, that decision based on either the specific nature of the problem, for example uncertainty in the user's ratings, or business policies, like the use of a pre-existing content-based system and know-how the company has invested on.

During the period content-based recommender systems were in their prime, the Java platform was not as popular as it is today. However, there are many content-based recommender systems built using Java technology. The use of the Java platform has transformed the way we build Web applications, including recommender systems. It allows us to develop large-scale Web applications in a single robust framework, without the need for individual tools, while providing tools for important operations, like connecting databases to our applications or providing secure services.With the use of Java technologies, like servlets, the JDBC library and the Java Security library, we are able to provide a unified service, which is easier to develop and maintain.

In the spirit of this paper, I am going to discuss the following systems :

1. LIBRA : A Content-Based Book Recommending System, using learning for text categorization.
2. CBMRS : A Content-Based Music Recommendation System.

A. LIBRA

LIBRA (Learning Intelligent Book Recommending Agent) is a content-based recommendation system that uses machine learning methodology in order to extract semi-structured text data from the web, for the purpose of making book recommendations [4]. The data extracted is relevant to a specific book and it is used to represent it in the system. LIBRA analyses the books' meta-data, in order to find books that correlate highly with the user's interests, thus making them a suitable candidate for a recommendation. LIBRA is operational in an experimental environment. The following link , http://www.cs.utexas.edu/users/libra/help.html provides more details in order to use LIBRA on-line.

On In order to populate the database, data for every book is extracted from the web pages of the Amazon.com web site. In order to identify the site related to every book, the Amazon subject search service is used. Therefore, instead of using the entire text contained in the book, we represent every item using textual meta-data, thus greatly reducing the input data size for the recommendation system. When this procedure is completed, the user provides a 1 to 10 rating for a selected set of books, which comprises the training data set the algorithm uses in order to construct a model of the user's interests. The user model is created using a Bayesian learning algorithm that represents the user's interest as a ranked list of the features that are common in highly rated titles that exist in the system's catalog. In the rest of this section we provide the details of the information extraction from the semi-structured data and the user's profile learning procedure.

As we've mentioned before, the system at first creates the database which will be used by performing an Amazon subject search to obtain a list of book-description URLs of relevant titles. After these pages are downloaded, information is extracted from the semi-structured data they contain by locating specific information we're interested in. This allows us to represent every book as structured data, rather than in its native unstructured text data. In order to represent a book, LIBRA finds a set of substrings (wrappers) from the document for each set of pre-specified slots. These slots contain the data which is going to be used by the content-based algorithm to provide the recommendations.The particular algorithm of LIBRA utilizes the slots of title, authors, synopses, published reviews, customer comments, related authors, related titles and subject terms. LIBRA requires that books with at least one synopsis, review or customer comment , in order to have enough information to produce an accurate result. The strings that are contained in each slot are separated into atomic words, also known as tokens, which are then inserted into an unordered bag of words. As a result, there are as many bags as there are slots. This process allows us to represent every book in a vector space, thus allowing for the application of most machine learning algorithms..

After the creation of the system's database, the user is required to rate a set of training books. This set is selected either manually by the user by searching for particular authors or by the system, automatically providing random titles from the database. The learning algorithm of the current system is a simple Bayesian text classifier , extended by the creators of LIBRA to handle multiple bags of words. Through the user's ratings the set of books is separated into two classes, positively rated books , which are books that received a rating from 6 to 10, and negatively rated books, which received a rating from 1 to 5. Those numerical ratings are used to assign weights to the training examples when we are building the model of the user's preferences. Furthermore we extend the Bayes classifier with Laplace estimates to avoid zero probabilistic results. Details of the theory supporting Bayes classifiers were provided in chapter 2 of this paper. After we have learned the user's profile we find a constant number of books or as noted "the top-k books" that correlate the most with his profile. Then, we can recommend this items to the user, using a proper illustration and description.

What differentiates LIBRA from other book recommender systems is the use of machine learning methods to provide input for the content-based system. In terms of performance. the computational complexity analysis of the system revealed that the complexity is linear to the input size, whereas experimental data was used to calculate an average throughput of 200 books per second in the process of creating the user's profile. Furthermore, an extensive statistical analysis illustrated in [4], provides strong evidence that the system achieves a high level of accuracy in its results, while presenting a satisfactory performance in terms of resources and computational time.

B. CBMRS

CBMRS is a system that implements the innovative idea of using not only the available data of the user's preferences but also dynamic data from the environment, in order to make music-related recommendations. The approach involves taking into consideration parameters such as weather, the user's pulses and mood, temperature and the user's current location and create a model based on how these parameters affect the user's preferences in music selection. By combining the current content-based filtering algorithms used for music recommendation and these new parameters that are known to affect the user's choice of music this system attempts to provide more accurate recommendations than the conventional content-based algorithms.

In this subsection we are going to discuss how the system collects the dynamic data from the enviroment and then uses it in combination with the static data collected from the user's preferences. Furthermore, we are going to discuss the general process of deciding which music titles to recommend to the user.

First of all, we will analyze how the system collects the data necessary to make a recommendation. For the purpose of measuring the user's pulses the system collects data from a pulse sensor attached to the user's watch. The sensor's signal is transmitted in real-time conditions using the Zigbee communication standard [5]. Then the data undergoes a process of discretization, into one of the following states: 0 to 40 is classified as dangerous, 41 to 65 as low, 66 to 120 as normal, 121 to 180 as high and 181 and up as dangerous.

In order to draw information about atmospheric conditions, the system collects weather data from the Web regarding the general area pinpointed from the user's IP. Then, this data is classified in seven distinct states for this variable, which are: Clear, sunny, cloudy, shower, rain, snow and storm. Furthermore, the atmosphere's temperature is measured, in order to provide additional information about atmospheric conditions. In order to collect this data, the system employs a temperature sensor attached on the user's watch. Then,this information is classified into one of the four distinct states for this variable: The temperature is considered cold when the readings are from -4 Fahrenheit degrees to 30.2 Fahrenheit degrees ,cool when it is in the range of 32 Fahrenheit degrees to 68 Fahrenheit degrees, warm from 69.8 Fahrenheit degrees to 86 Fahrenheit degrees and hot when the temperature is from 87.8 Fahrenheit degrees and up.

To pinpoint the user's location the system uses an RFID (Radio Frequency Identification) Tag also attached to the user's watch. This variable has six distinct states: Balcony, Bathroom, Bedroom, Guestroom, Kitchen and Living Room.

The age of the user can be traced using the RFID tag previously mentioned. Five distinct states are used for this variable: 0 to 7:Infant, 8 to 11:Child, 12 to 17:Young Adult, 18 to 61:Adult and 62 and up:Old Adult. There also exists a variable for the sex of the user which has only two distinct values and is traced using the

RFID Tag.

The system collects statistic data about the user's choices, referred to as static data in contrary to the dynamic data provided by the previously mentioned sensors, the reason being that it applies to any state of the environment. The recommendation algorithm must be executed while the dynamic data collected from the environment has not changed, in order to provide accurate recommendations. This soft real-time restriction allows the system to make an estimate of the type of music the user would prefer to listen while certain environment variables apply, matching the mood of the user with his music preferences. After the system has stored the dynamic data concerning the state of the user's environment, this information is combined with the user's profile, which uses statistical data to model the user's preferences in music genre and style. Then, it recommends songs that not only match his interest but also the conditions of his current environment. A constant number of songs are recommended, referred to as "the top-k results".

The system is based on a Java-Based OSGi framework. The system was developed based on an OSGi gateway using Knopflerfish 1.3.3 [6], an open architecture source project which implements a service framework. OSGi is an industrial standard which is used to connect several different internet devices such as household information devices and security systems. It is a JES-based Gateway software based on an open architecture Java embedded server which can provide high quality multimedia regardless of the application software platform. It is an open network architecture that can support various network physical mediums and protocols (USB, Bluetooth, PNA, HAVi, RF, VESA etc.). As we have mentioned previously, the system consists of many different devices interconnected to each other providing data to the recommendation system, therefore an open architecture standard like OSGi is needed to make the interconnection faster and easier.

The CBMRS system, from the viewpoint of recommender systems, implements a content-based recommendation algorithm. We will describe not only the algorithm but also the process which the systems follows and how the different modules of the system interoperate in this process. There are three distinct modules in the CBMRS: the Context Manager, the Service Manager and the Music Recommendation Manager.

The Context Manager transfers data generated by events that change the state of the system. The change of state is caused by a new measurement on the sensors collecting the context data, which is then processed by a context analyzer. The analyzer's output is then transfered to an OWL (Web Ontology Language) inference engine [7]. In this process, OWL's purpose is to discover and represent the semantics and relationships that characterizes the system's data. After the data has been processed by the OWL inference engine, the data is sent to the Service Manager where it is transformed into valuable information using an OWL inferencer, including an OWL ontology object DB. The Inference engine in the Context Manager uses a Jena 2.0 ontology inferencer.

The second part of the architecture is the Service Manager. It consists of a Bundle Service, a term used in OSGi to refer to a collection of services, that provides the following services: a recommendation service as a bundle in a SOAP(Simple Object Access Protocol) [8] service, an OSGi framework installed device used to transfer information received from the OWL inference engine to the recommendation system and an Application and Bundle Manager Service that supports the management of the mobility of bundles. Communication between the Context Manager, the Music Recommendation Manager and the Music Service is performed using the SOAP service to process in real-time the data needed to offer music recommendations, using context information from as many different systems as possible, without interruption by a different device that has an OSGi middleware even when a user is on moving to a different location.

The third part of this architecture, the Music Recommendation Manager plays the role of deciding the optimal music list in a recommendation module. This is achieved by combining the data received from the recommendation list, which corresponds to a certain context information received from the Service Manager, with the user profile and information stored in the MCIDB (Music Content Information DataBase). This data is then used provided as input to a filtering process in a recommendation module. The Query Manager selects a proper profile according to the user context information and transfers the related data to the Recommendation Module. It also performs an updating process for the music selected by the users in the User Profile, further improving the statistical profile created for the current user. Then, the Recommendation Module performs a token analysis of the profile received from the Query Manager and configures a recommendation list by searching music from the MCIDB. Having created our recommendations, we can present them to the user in the form of a song playlist.

Summarizing, CBMRS uses a real-time open architecture implemented using Java, in which several different devices are used in order to obtain information critical to the system's accuracy. This information is then combined with information about the user's interest and a music related database in order to make a recommendation. Although the system is comprised of a rather complex architecture and uses many different sources of data, it allows us to make music recommendations to the user not only dependent on his preferences, but also taking into consideration his current mood, as well as other data. Therefore, although CBMRS is a soft real-time system, with strict performance requirements, it provides a service to users which was unavailable up to now.

➢ 4.4 Text Vectorization

Text Vectorization is the process of converting text into numerical representation. Here is some popular methods to accomplish text vectorization:

- Binary Term Frequency
- (L1) Normalized Term Frequency
- Bag of Words (BoW) Term Frequency

Binary Term Frequency

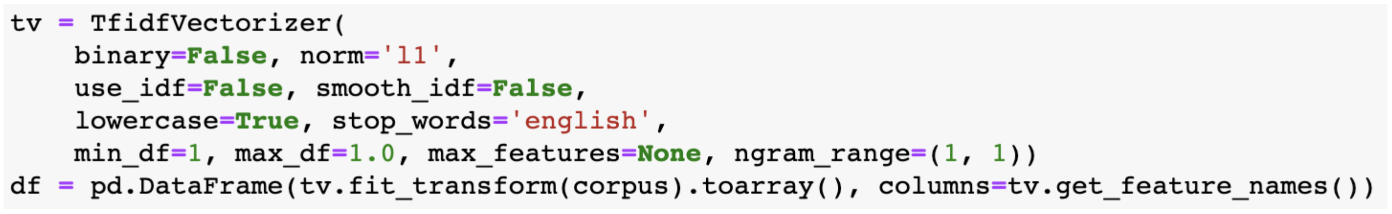
Binary Term Frequency captures presence (1) or absence (0) of term in document. For this part, under TfidfVectorizer, we set binary parameter equal to true so that it can show just presence (1) or absence (0) and norm parameter equal to false.

```
tv = TfidfVectorizer(
    binary=True, norm=False,
    use_idf=False, smooth_idf=False,
    lowercase=True, stop_words='english',
    min_df=1, max_df=1.0, max_features=None, ngram_range=(1, 1))
df = pd.DataFrame(tv.fit_transform(corpus).toarray(), columns=tv.get_feature_names())
```

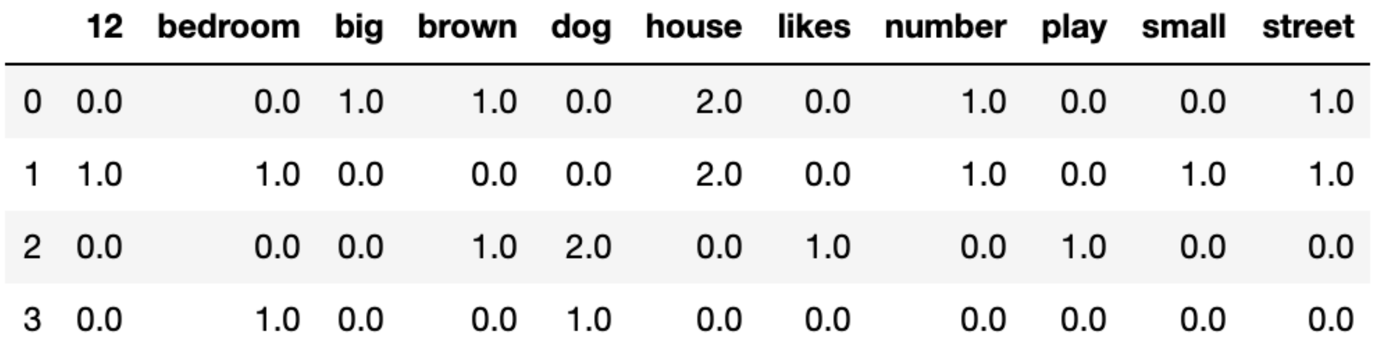
	12	bedroom	big	brown	dog	house	likes	number	play	small	street
0	0.0	0.0	1.0	1.0	0.0	1.0	0.0	1.0	0.0	0.0	1.0
1	1.0	1.0	0.0	0.0	0.0	1.0	0.0	1.0	0.0	1.0	1.0
2	0.0	0.0	0.0	1.0	1.0	0.0	1.0	0.0	1.0	0.0	0.0
3	0.0	1.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0

(L1) Normalized Term Frequency

(L1) Normalized Term Frequency captures normalized BoW term frequency in document. Under TfidfVectorizer, we set binary parameter equal to false so that it can show the actual frequency of the term and norm parameter equal to l1.



```
tv = TfidfVectorizer(
    binary=False, norm='l1',
    use_idf=False, smooth_idf=False,
    lowercase=True, stop_words='english',
    min_df=1, max_df=1.0, max_features=None, ngram_range=(1, 1))
df = pd.DataFrame(tv.fit_transform(corpus).toarray(), columns=tv.get_feature_names())
```



	12	bedroom	big	brown	dog	house	likes	number	play	small	street
0	0.0	0.0	1.0	1.0	0.0	2.0	0.0	1.0	0.0	0.0	1.0
1	1.0	1.0	0.0	0.0	0.0	2.0	0.0	1.0	0.0	1.0	1.0
2	0.0	0.0	0.0	1.0	2.0	0.0	1.0	0.0	1.0	0.0	0.0
3	0.0	1.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0

Bag of Words (BoW) Term Frequency

A bag-of-words model, or BoW for short, is a way of extracting features from text for use in modeling, such as with machine learning algorithms.

The approach is very simple and flexible, and can be used in a myriad of ways for extracting features from documents.

A bag-of-words is a representation of text that describes the occurrence of words within a document. It involves two things:

1. A vocabulary of known words.
2. A measure of the presence of known words.

It is called a “bag” of words, because any information about the order or structure of words in the document is discarded. The model is only concerned with whether known words occur in the document, not where in the document.

Bag of Words (BoW) Term Frequency captures frequency of term in document. Under TfidfVectorizer, we set binary parameter equal to false so that it can show the actual frequency of the term and norm parameter equal to none.

The intuition is that documents are similar if they have similar content. Further, that from the content alone we can learn something about the meaning of the document.

The bag-of-words can be as simple or complex as you like. The complexity comes both in deciding how to design the vocabulary of known words (or tokens) and how to score the presence of known words.

Limitations of Bag-of-Words

The bag-of-words model is very simple to understand and implement and offers a lot of flexibility for customization on your specific text data.

It has been used with great success on prediction problems like language modeling and documentation classification.

Nevertheless, it suffers from some shortcomings, such as:

- Vocabulary: The vocabulary requires careful design, most specifically in order to manage the size, which impacts the sparsity of the document representations.
- Sparsity: Sparse representations are harder to model both for computational reasons (space and time complexity) and also for information reasons, where the challenge is for the models to harness so little information in such a large representational space.
- Meaning: Discarding word order ignores the context, and in turn meaning of words in the document (semantics). Context and meaning can offer a lot to the model, that if modeled could tell the difference between the same words differently arranged (“this is interesting” vs “is this interesting”), synonyms (“old bike” vs “used bike”), and much more.

Chapter-5
Code & Dataset

DATASET:

A data set (or dataset) is a collection of data. In the case of tabular data, a data set corresponds to one or more database tables, where every column of a table represents a particular variable, and each row corresponds to a given record of the data set in question.

For the project we have utilized the movie dataset from ‘TMDB 5000 Movie Dataset’ (Kaggle).

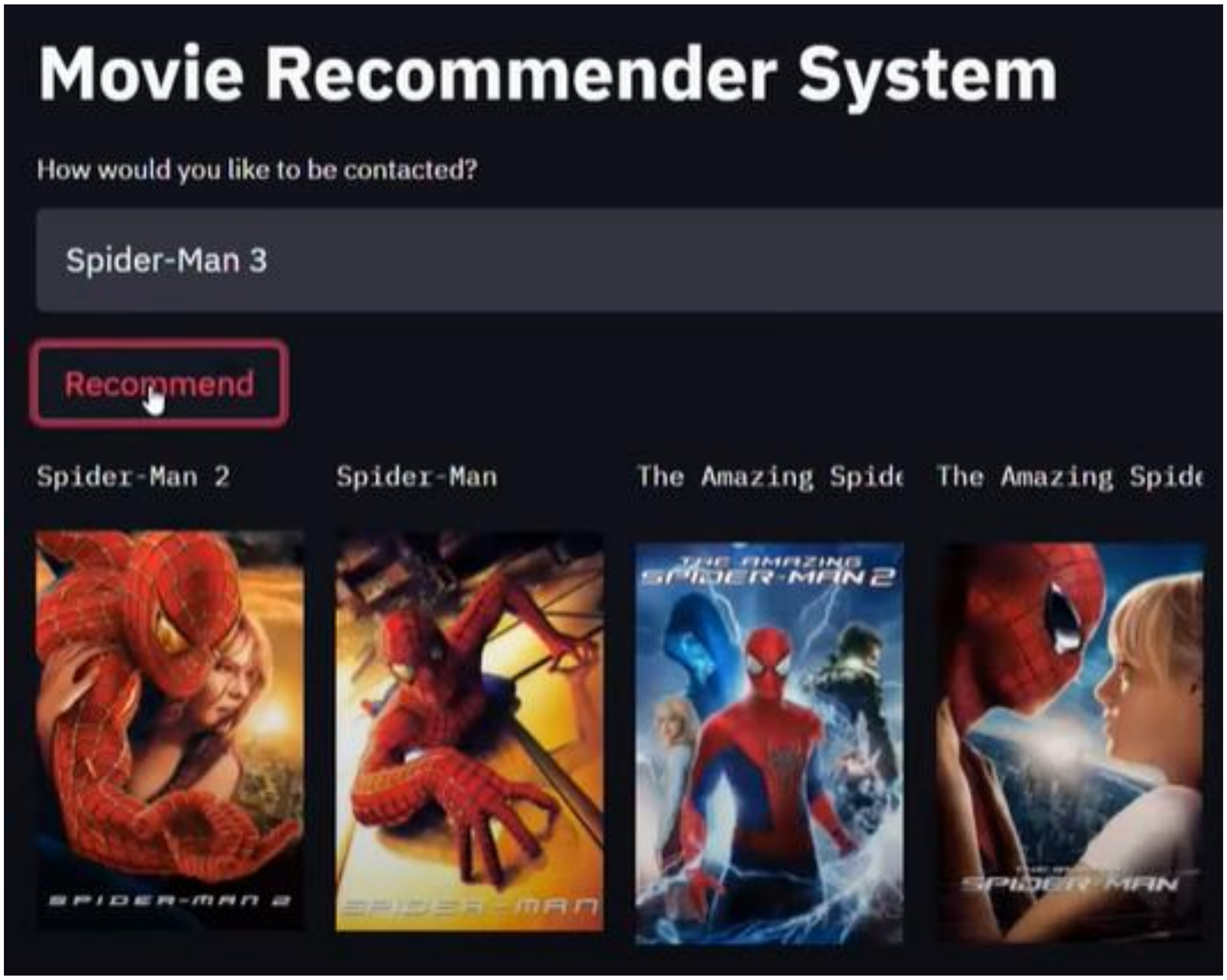
- TMDB 5000 Movie contain data of 5000. movies
- Along with recommendation user grt movie poster.
- All users get 5 recommendation

Python code

```
import
stream
lit as st
        import pickle
        import pandas as pd
        import requests
        def fetch_poster(movie_id):
            response =
        requests.get('https://api.themoviedb.org/3/movie/{}?api_key=dc59d36d850605fd3587d94abcca8
        3bd&language=en-US'.format(movie_id))
            data = response.json()
            return "https://image.tmdb.org/t/p/w500/" + data['poster_path']
        def recommend(movie):
            movie_index = movies[movies['title'] == movie].index[0]
            distances = similarity[movie_index]
            movies_list = sorted(list(enumerate(distances)), reverse=True, key=lambda x: x[1])[1:6]
            recommended_movies = []
            recommended_movies_posters = []
            for i in movies_list:
                movie_id = movies.iloc[i[0]].movie_id
                recommended_movies.append(movies.iloc[i[0]].title)
                # fetch poster from API
                recommended_movies_posters.append(fetch_poster(movie_id))
            return recommended_movies,recommended_movies_posters
        movies_dict = pickle.load(open('movie_dict.pkl','rb'))
        movies = pd.DataFrame(movies_dict)
        similarity = pickle.load(open('similarity.pkl','rb'))
        st.title('Movie Recommender System')
        selected_movie_name = st.selectbox(
        'Please select a Movie',
        movies['title'].values)
        if st.button('Recommend'):
            names,posters = recommend(selected_movie_name)
            col1, col2, col3, col4, col5 = st.beta_columns(5)
            with col1:
                st.text(names[0])
                st.image(posters[0])
            with col2:
```

```
    st.text(names[1])
    st.image(posters[1])
with col3:
    st.text(names[2])
    st.image(posters[2])
with col4:
    st.text(names[3])
    st.image(posters[3])
with col5:
    st.text(names[4])
    st.image(posters[4])
```

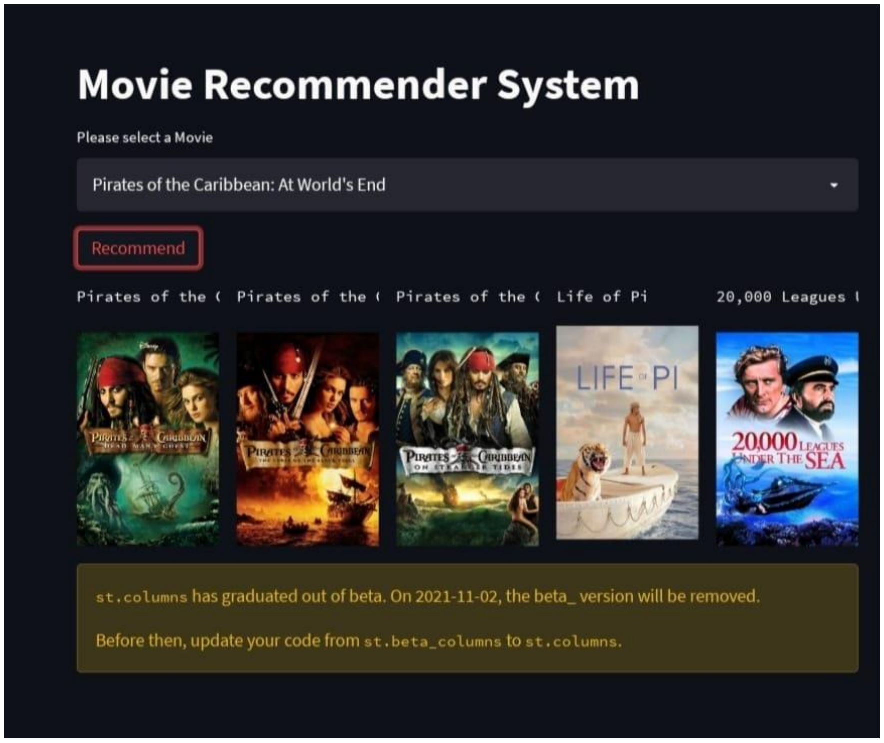
APPLICATION



Movie Recommender System

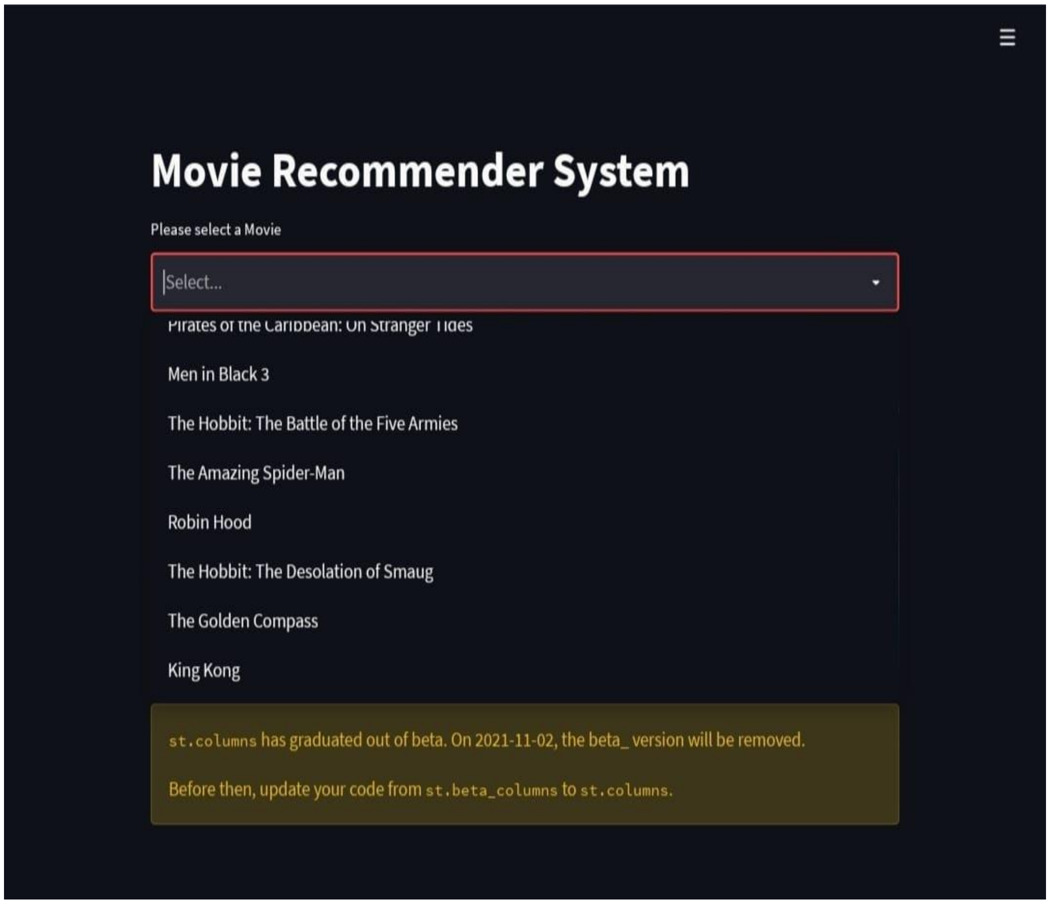
How would you like to be contacted?

Spider-Man 3



st.columns has graduated out of beta. On 2021-11-02, the beta_ version will be removed.

Before then, update your code from st.beta_columns to st.columns.



Movie Recommender System

Please select a Movie

Select...

Pirates of the Caribbean: On Stranger Tides

Men in Black 3

The Hobbit: The Battle of the Five Armies

The Amazing Spider-Man

Robin Hood

The Hobbit: The Desolation of Smaug

The Golden Compass

King Kong

`st.columns` has graduated out of beta. On 2021-11-02, the beta_ version will be removed.

Before then, update your code from `st.beta_columns` to `st.columns`.

Chapter-6
Conclusion and Future Scope

Conclusion

Recommender system has become more and more important because of the information overload. For content-based recommender system specifically, we attempt to find a new way to improve the accuracy of the representative of the movie.

Proper implementation of this project can lead to time saving of inexperienced people by providing them with movie recommendations from our TMDB 5000 Movie Dataset

Future Scope

Currently, there are many fields where recommendation-based services are used such as stock price predictor tools used by stock brokers. Therefore, there is requirement for service like this in the movie industry which can help the customers in getting recommendation. There are many researches works that have been done on this using various techniques and more research is needed to improve the accuracy of the prediction by using different algorithms. More accurate data with better features can be also be used to get more accurate results.

The recommender system has been evolving for many years and has just reached a low point. Machine learning, large-scale networks, and high-performance computers have all accelerated significant developments in this subject in recent years. In our future efforts, we shall take into account the following factors.

Introduce machine learning:
For future study, dynamic parameters will be introduced into recommender system, we will use machine learning to adjust the weight of each feature automatically and find the most suitable weights.

collaborative filtering recommendation:
After getting enough user data, collaborative filtering recommendation will be introduced. As we discussed in Section above, collaborative filtering is based on the social information of users, which will be analyzed in the future research.

Sentimental Analysis:
Along with movie recommendations , we can use sentimental analysis on reviews of the movie. So that we can show a user, negative and positive reviews of the movie.

References

[1] Gediminas Adomavicius and Alexander Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. Knowledge and Data Engineering, IEEE Transactions on, 17(6):734–749, 2005.

[2] Suvir Bhargav. Efficient features for movie recommendation systems. 2014.

[3] Robin Burke. Hybrid web recommender systems. In The adaptive web, pages 377–408. Springer, 2007.

[4] Scott C. Deerwester, Susan T Dumais, Thomas K. Landauer, George W. Furnas, and Richard A. Harshman. Indexing by latent semantic analysis. JAsIs,41(6):391–407, 1990.

[5] Greg Linden, Brent Smith, and Jeremy York. Amazon. com recommendations:Item-to-item collaborative filtering. Internet Computing, IEEE, 7(1):76–80, 2003.

[6] Xu Hailing, Wu xiao, Li Xiaodong, and Yan Baoping. Comparison study of internet recommendation system. Journal of Software, 20(2):350–362, 2009.

[7] Pasquale Lops, Marco De Gemmis, and Giovanni Semeraro. Content-based recommender systems: State of the art and trends. In Recommender systems handbook, pages 73–105. Springer, 2011.

[8] Dietmar Jannach, Markus Zanker, Alexander Felfernig, and Gerhard Friedrich. Recommender systems: an introduction. Cambridge University Press, 2010.