A

Project Report

On

Book Recommendation System

By

Satyam Pant(1613310190)
Sahil Pandey(1613310184)

Under the Supervision of

Mr. Kedarnath Singh
Assistant Professor (CSE Dept)

Submitted to the department of Computer Science and Engineering

For the partial fulfillment of the requirements

for award of Bachelor of Technology

in

Computer Science and Engineering

Noida Institute of Engineering & Technology Gr. Noida
Dr. A.P.J. Abdul Kalam Technical University, Lucknow, Uttar Pradesh, India
May, 2019-2020

Certificate

This is to certify that the Project report entitled **"Book Recommendation System"** is a record of the work done by the following students:

Student name	**Roll No.**
Satyam Pant	1613310190
Sahil Pandey	1613310184

This work is done under my/our supervision and guidance during the academic year of 2019-20.This report is submitted to the **Noida Institute of Engineering & Technology, Greater Noida** for partial fulfillment for the degree of **B.TECH. (Computer Science and Engineering)** of **Dr A P J Abdul Kalam Technical University, Lucknow, Uttar Pradesh, India.**

I/We wish him/her all the best for all the endeavors.

Signature of Guide:
Mr. Kedarnath Singh
Assistant Professor

ACKNOWLEDGEMENT

I would like to place on record my deep sense of gratitude to **Mr. Kedarnath Singh, Assistant Professor, Department of Computer Science and Engineering, Noida Institute of Engineering & Technology**, Greater Noida, Gautam Budh Nagar, Uttar Pradesh, India for his generous guidance, help and useful suggestions.

I express my sincere gratitude to **Prof. Chandra Shekhar Yadav, HOD CSE**, Noida Institute of Engineering & Technology, Greater Noida for his stimulating guidance, continuous encouragement and supervision throughout the course of present work.

Include other names as per your need in same manner

Date: **Student Name:**

05/05/2020

Satyam Pant

Sahil Pandey

ABSTRACT

Recommendation System (RS) is software that suggests similar items to a purchaser based on his/her earlier purchases or preferences. RS examines huge data of objects and compiles a list of those objects which would fulfil the requirements of the buyer. Nowadays most ecommerce companies are using Recommendation systems to lure buyers to purchase more by offering items that the buyer is likely to prefer. Book Recommendation System is being used by Amazon, Barnes and Noble, Flipkart, Goodreads, etc. to recommend books the customer would be tempted to buy as they are matched with his/her choices. The challenges they face are to filter, set a priority and give recommendations which are accurate. RS systems use Collaborative Filtering (CF) to generate lists of items similar to the buyer's preferences. Collaborative filtering is based on the assumption that if a user has rated two books then to a user who has read one of these books, the other book can be recommended (Collaboration). CF has difficulties in giving accurate recommendations due to problems of scalability, sparsity and cold start. Therefore this paper proposes a recommendation that uses Collaborative filtering with Jaccard Similarity (JS) to give more accurate recommendations. JS is based on an index calculated for a pair of books. It is a ratio of common users (users who have rated both books) divided by the sum of users who have rated the two books individually. Larger the number of common users higher will be the JS Index and hence better recommendations. Books with high JS index (more recommended) will appear on top of the recommended books list. Keywords: Similarity index, filtering techniques, recommender system.

TABLE OF CONTENTS

		Page No.
Certificate		i
Acknowledgement		ii
Abstracts		iii
Table of Contents		iv
CHAPTER 1	**INTRODUCTION**	**1-3**
	1.1 Architecture of the system	**2-3**
CHAPTER 2	**LITERATURE REVIEW/SURVEY**	**4-5**
CHAPTER 3	**PROPOSED SYSTEM**	**6-10**
CHAPTER 4	**WORKING**	**11-22**
	4.1 Collection of data	11
	4.2Data Overview	11-13
	4.3 Data Cleaning	13-15
	4.4 Data Analysis	15-17
	4.5 Data Preparation	17-19
	4.6 Modelling	19-21
	4.7 User Similarities and Recommendations	21-22
	4.8 Answering Questions	22
CHAPTER 5	**WHAT ARE THE DIFFERENT TYPES OF RECOMMENDATIONS?**	**23-26**
	5.1 Collaborative Filtering	23-24

	5.2 Content Based Filtering	24-25
	5.3 Hybrid Filtering	25-26
CHAPTER 6	**FEASIBILITY STUDY**	**27**
CHAPTER 7	**SYSTEM REQUIREMENTS**	**28**
	7.1 Hardware Components	28
	7.2 Software Requirements	28
CHAPTER 8	**ADVANTAGE AND DISADVANTAGE**	**29-31**
CHAPTER 9		**31-42**
		31
CHAPTER 10	**PROBLEM ANALYSIS AND SOLUTION**	**31-39**
		40
	9.1 Problem Analysis	**40-42**
	9.1.1 The Challenge	
	9.2 Solution	**43**
	9.2.1 Design choices for the recommender engine and integration strategy	**40-42**
CHAPTER 11	**EXPERIMENTAL RESULTS**	**43**
CHAPTER 12	**CONCLUSION AND FUTURE WORK**	**44**
CHAPTER 13	**REFERENCES**	**45-46**

LIST OF FIGURES

Figure No.	Figure Name	Page No.
Figure 1.1	Physical Architecture of the System	2
Figure 1.2	Overview of the Components of the Hybrid Recommender System	3
Figure 3.1	Login Module for the Proposed System	7
		8
Figure 3.2	Search Results	9
Figure 3.3	Related Books	
Figure 3.4	Block Diagram	9
Figure 3.5	Architecture of Proposed System	10
Figure 4.1	Times Book Rated	16
Figure 4.2	Reviewers rated the same title	16
Figure 5.1	Types of Filtering	24
Figure5.2	Hybrid Recommendation	26
Figure10.1	Comparison of Ratings and Predictions for user-item pairs	43

LIST OF TABLES

Table No.	Table Name	Page No.
Table .1	Data Head	11
Table 4.2	Modified Data Head(1)	12
Table 4.3	Modified Data Head(2)	14
Table 4.4	Data Preparation	17

CHAPTER 1
INTRODUCTION

Recommendation system filters information by predicting ratings or preferences of consumers for items that the consumer would like to use . It tries to recommend items to the consumer according to his/her needs and taste. RS mainly uses two methods to filter information - Content-based and Collaborative filtering. Content-based filtering involves recommending those items to a consumer which are similar in content to the items that have already been used by him/her. First, it makes a profile of the consumer, which consists of his/her taste. Taste is based on the type of books rated by the consumer. The system analyses the books that were liked by the consumer with the books he had not rated and looks for similarity. Out of these unrated books, the books with the maximum value of similarity index will be recommended to the consumer. Paul Resnick and Hal Varian were the ones who suggested Collaborative filtering algorithm in 1997. It became popular amid the various frameworks available at that time.

A complete RS contains three main things: user resource, item resource and the recommendation algorithm. In the user model, the consumers' interests are analysed, similarly, the item model analyses the items' features. Then, the characteristics of the consumer are matched with the item characteristics to estimate which items to recommend using the recommendation algorithm. The performance of this algorithm is what affects the performance of the whole system.

In memory-based CF, the book ratings are directly used to assess unknown ratings for new books. This method can be subdivided into two ways: User-based approach and Item-based approach.

1. User-based approach: As per this method, customers with alike choices form a neighbourhood. If an item is not rated by the buyer, but it has been rated highly by other members of the neighbourhood, then it can be endorsed to the buyer. Hence the buyer’s choices can be predicted based on the neighbourhood of similar buyers.

2. Item-based approach: In this method, similarity between the group of objects rated emphatically by the buyer and the required object is calculated. The items which are very alike are selected. Recommendation is computed by finding the weighted means of user’s ratings of the alike objects. To obtain accurate and faster recommendations

researchers have combined the different recommender technologies, these are known as Hybrid recommendation systems. Some Hybrid recommendation approaches are: • Separate execution of procedures and connecting the results. • Using some content filtering guidelines with community CF. • Using some principles of CF in content filtering recommender • Using both Content and Collaborative filtering in a recommender Additionally there is on-going research on Semantic-based, Context-aware, Cross-lingual, Crossdomain and peer-to-peer methods.

1.1. Architecture of the system

Architecture is a set of structuring principles that enables a system to be comprised of a set of simpler systems each with its own local context that it is independent of but not inconsistent with the context of the larger system as a whole. Architecture is created to describe the structure of the system to be built and how that structure supports the business and service-level requirements. In this section we are going to discuss the components the system is made up of.

Before we discuss every single component of the system we first want to give an overview of the physical architecture of the system, as well as an overview of its logical architecture. Note that this is a minimalist architecture and does not impose any implementation method. In fact both servers could be physically implemented in a single network node.

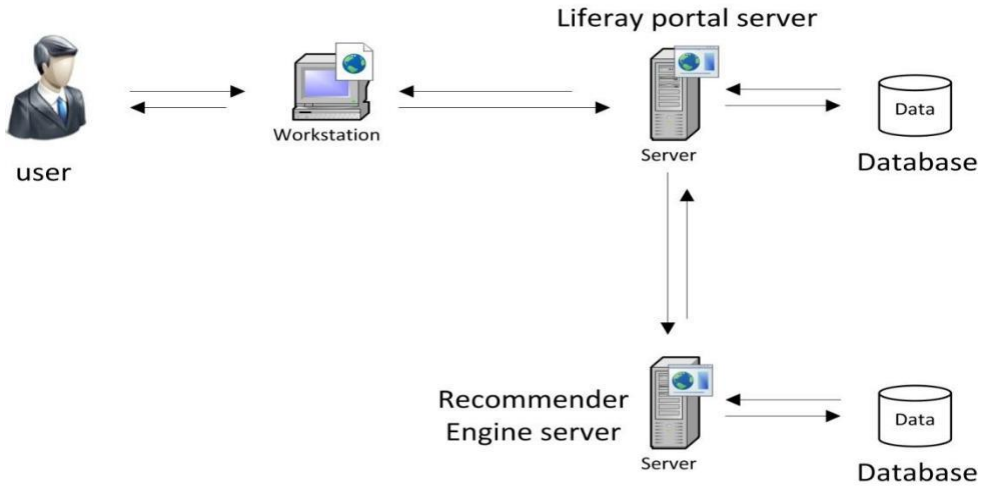

Fig1.1.Physical architecture of the system

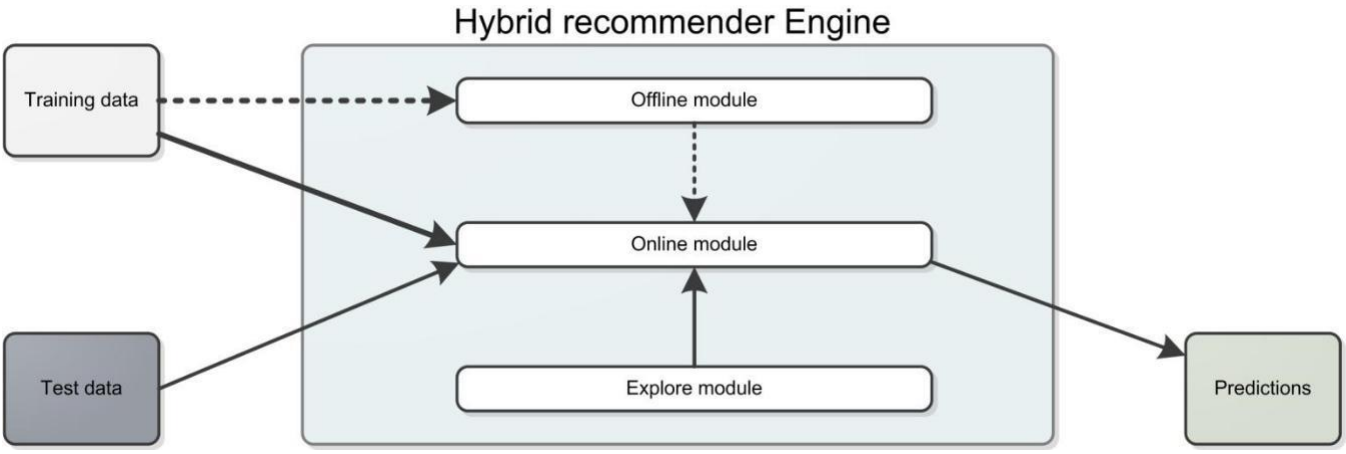

Fig1.2.Overview of the components of the hybrid recommender engine

The above figure displays the global architecture of the hybrid recommender system. Content and collaborative data are aggregated and feature matrices are computed by the offline module. The aggregation process yields unprocessed collaborative data to the online module. The online module processes the collaborative data and uses it to predict users' preferences. The explore module builds lists of items that must be explored by the user, and in adequate time passes them to the online module.

CHAPTER 2
LITERATURE REVIEW/SURVEY

Okon et.al. (2018) proposed a model that generates recommendations to buyers, through an enhanced CF algorithm, a quick sort algorithm and Object Oriented Analysis and Design Methodology (OOADM). Scalability was ensured through the implementation of Firebase SQL. This system performed well on the evaluation metrics.

Kurmashov et.al. (2015) used Pearson correlation coefficient based CF to provide internet based recommendations to book readers and evaluated the system through an online survey.

Mathew et.al. (2016) proposed a system that saves details of books purchased by the user. From these Book contents and ratings, a hybrid algorithm using collaborative filtering, content-based filtering and association rule generates book recommendations. Rather than Apriori, they recommended the use of Equivalence class Clustering and bottom up Lattice Transversal (ECLAT) as this algorithm is faster due to the fact that it examines the entire dataset only once.

Parvatikar et.al. (2015) proposed item-based collaborative filtering and association rule mining to give recommendations. Similarity between different users was computed through Adjusted Cosine Vector Similarity function. Better recommendations were obtained as through this method data sparsity problem was removed.
Ayub et.al. (2018) proposed a similarity function similar to Jaccard Similarity to locate alike items and users for the enquiring item and user in nearest neighbour based collaborative filtering. They proposed that absolute value of ratings should be taken as against the ratio of co-rated items taken in Jaccard Similarity. They also compared performance of their method with other similarity measures.

Gogna and Majumdar suggested the use of buyer's demographic and item category to overcome data sparsity and cold start problems in their movie recommendation system. Latent Factor Model (LFM) was used. They developed a matrix to match the buyer and user information to get a dense user and dense item matrix. Label Consistency map, the outcome of this system was used to suggest unrated and other items to new buyers.

Chatti et.al. (2013) suggested tag-based and rating based CF recommendation in technology enhanced learning (TEL) to resolve the data sparsity problem and extract relevant information from the rating database. Memory and model oriented 16 varied tag-based Collaborative filtering algorithms were evaluated for buyer satisfaction and accuracy of recommendations in Personal Learning Environments.
Choi et.al. (2010) proposed RS based on HYRED, a hybrid algorithm using both content and collaborative filtering on a compact dataset (by reducing user interest items) and neighbor data. HYRED used altered Pearson Coefficient based Collaborative filtering and distance-to-boundary (DTB) Content filtering. This would result in better and faster recommendation for large amount of data.

Liu et.al. (2012) added the dimension of user-interest. They proposed iExpand, a 3 tier model i.e. user, user-interest and item. This helped in overcoming the issues of overspecialization and cold-start as well as reducing computation costs.

Feng et.al. (2018) proposed a RS for movies based on a similarity model constituted of factors S1 (similarity between users), S2 (ratio of co-rated items) and S3 (user's rating choice weight). This RS was particularly useful for sparse datasets.

Literature survey suggested that recommendation systems are being used by large number of online marketers to increase their sales by offering products to customers which match their tastes.

These RS suffer from many problems such as data sparsity, cold start, trust, scalability and privacy.
Therefore there is need for improved recommendation systems which solve these issues.

Most researchers use Adjusted Cosine Vector Similarity function to compute similarity among book ratings to recommend books, to take into account that users have different rating schemes. Some rate items high usually while others usually rate low.

CHAPTER 3
PROPOSED SYSTEM

According to Aggarwal, C.C., Collaborative Filtering method is used in recommendation systems to develop recommendations based on ratings provided by the other users of the system [17]. If buyer's ratings of items match, it is likely that their ratings of other items will also match, this is the basic assumption of CF. Computers cannot gauge qualitative factors such as taste or quality, therefore recommendations based on the ratings of humans who can rate on the basis of qualitative factors, i.e. collaboration, will give a better outcome.
Soanpet Sree Lakshmi and Dr. T. Adi Lakshmi in their work have described in detail the problems like overspecialization, data sparsity, cold start, scalability, ranking of the recommendation, etc. that are related to the CF recommendation system [6]. Buyers do not like to spend too much time on rating items online. Hence rating data is usually sparse, which in turn reduces the recommendation quality. As new users have not made any explicit or implicit ratings, therefore it is difficult to find similarity and hence provide recommendations. This is known as cold start problem.

This paper proposes the use of Jaccard Similarity [11]. Jaccard Similarity for item-based recommendation calculates the similarity between two books by taking the number of users who have rated both books in the numerator and the number of users who have rated either of the two books in the denominator. This does not consider the absolute ratings rather it considers number of items rated. Two items will be more similar, when they have been rated by more common users. Jaccard Similarity does not consider the actual rating given by the user to a book, but only goes by the number of users who have rated the books. Incorporating a weightage to the similarity index based on the rating given by user, overcomes this issue. For the item-based algorithm, Jaccard Similarity is given as

$$(,) = \frac{| \cap |}{| \cup |}$$

Where is the set of users who have rated item A and is the set of users who have rated item B.

In this approach, the recommendation system will calculate the similarity between books that have been rated by the user already and the books available in the Book Crossing

dataset. This dataset was compiled by Cai-Nicolas Ziegler, it contains data of 2,71,379 books, along with 11,49,780 user ratings for these books, given by 2,78,858 users. Datasets are available in CSV (Comma Separated Values)
and SQL formats. Since large amount of data is to be handled, SQL database was used for faster and more efficient processing.

Login module for the proposed system:

Fig3.1 Login module for the proposed system:

Once the similarities have been computed for all the books, they are sorted and the books most similar to the books rated by the user are recommended to him. Recommendations to the user can also be based on a particular book chosen by the user. In this case, the similarity calculation will be done only for this book and the most similar books will be recommended to him. Books in the entire dataset can be searched by author or by title, and the user can give ratings to these books for more accurate results.

Fig3.2 Search Results

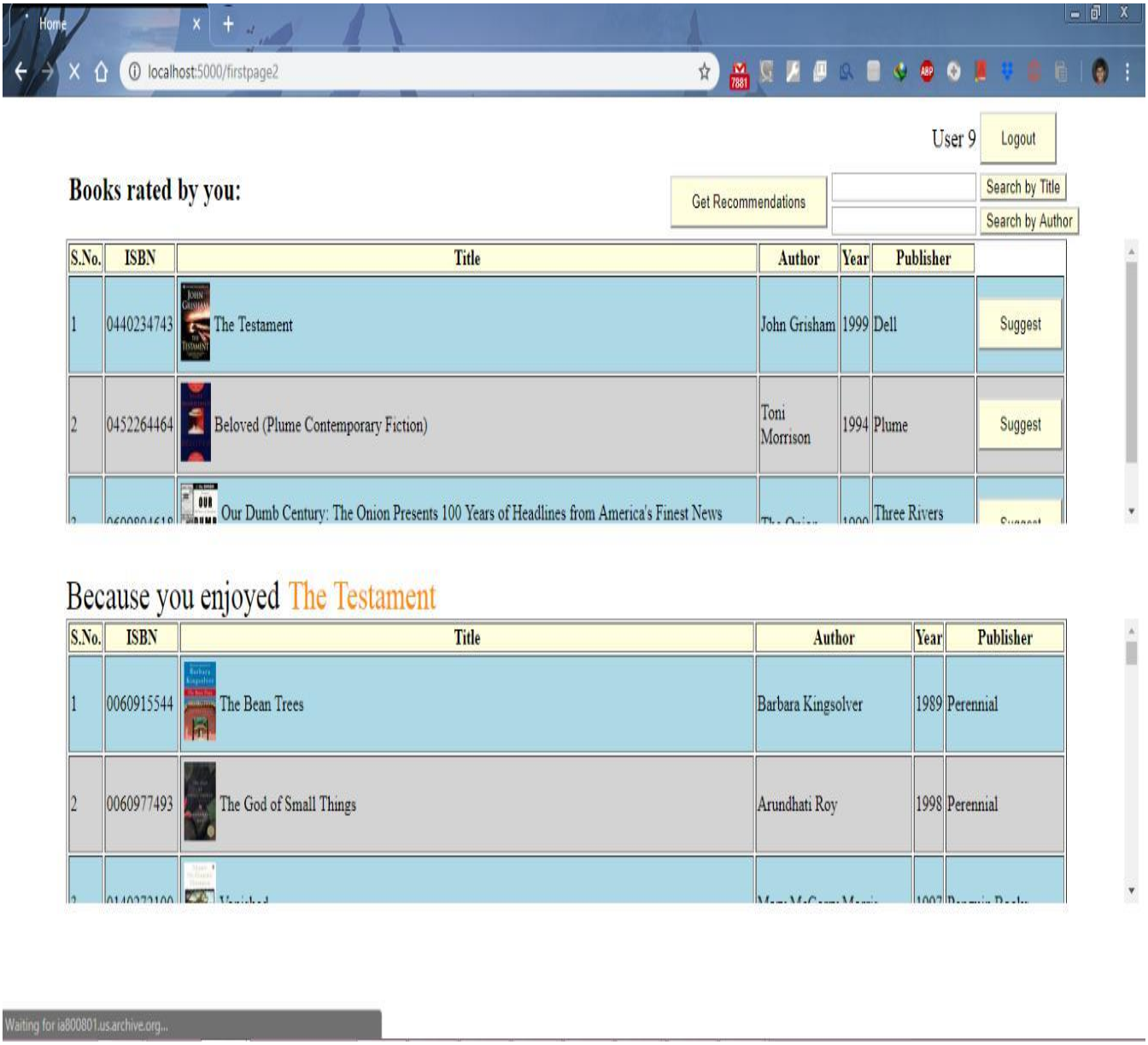

Fig3.3 Rated Books

The interaction between the application server and the user is shown in fig 1. Fig. 2 displays the architecture of the proposed system

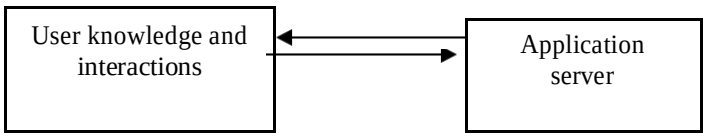

Fig 3.4 Block Diagram

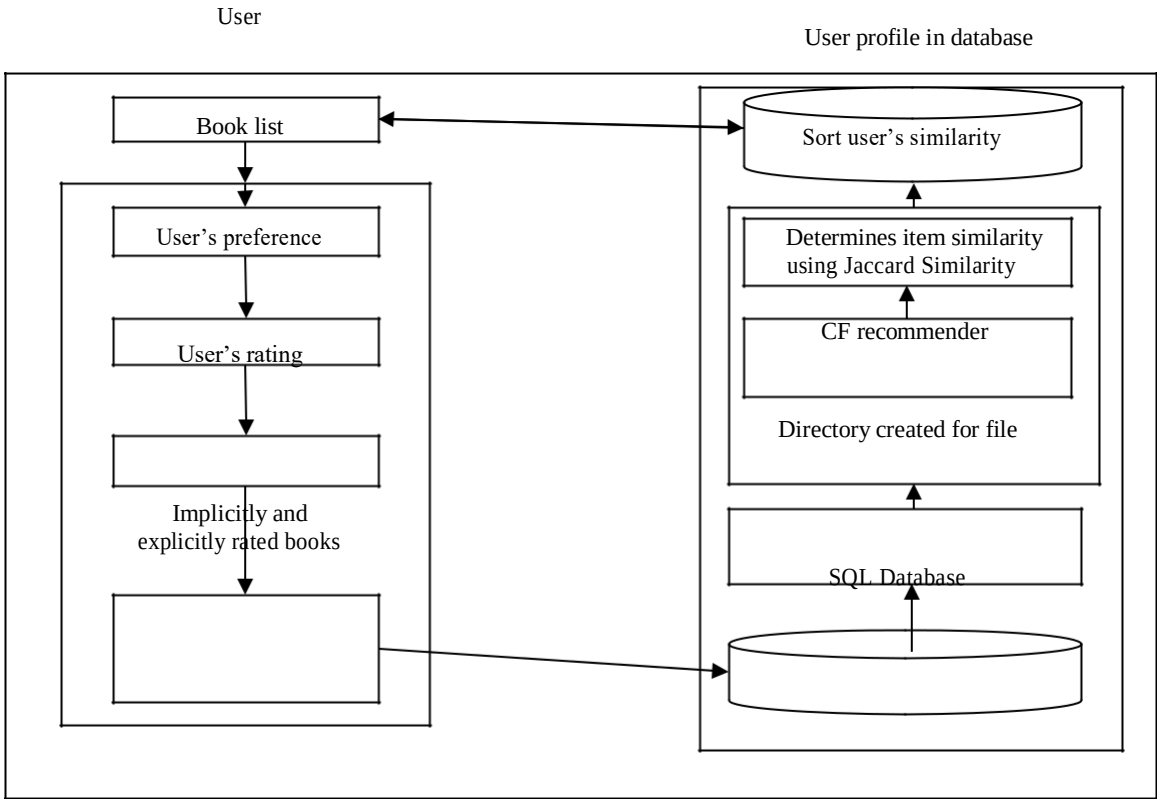

Fig. 3.5 Architecture of proposed system

CHAPTER 4

WORKING

According to the article Using Machine Learning on Compute Engine to Make Product Recommendations, a typical recommendation engine processes data through the following four phases namely collection, storing, analyzing and filtering.

Collection of data:

The first step in creating a recommendation engine is gathering data. Data can be either explicit or implicit data. Explicit data would consist of data inputted by users such as ratings and comments on products. And implicit data would be the order history/return history, Cart events, Pageviews, Click thru and search log. This data set will be created for every user visiting the site.

Data Overview:

```
import pandas as pd
csv = "br.csv"
df = pd.read_csv(csv, engine='python', error_bad_lines=False)
```

```
df.head(20)
```

```
Skipping line 312075: unexpected end of data
```

	bookID	title	author	rating	ratingsCount	reviewsCount	reviewerName	reviewerRatings	review
0	9	Unauthorized Harry Potter Book Seven News: "Ha...	W. Frederick Zimmerman	3.73	22	1	Charles G	3.0	NaN
1	8	Harry Potter Boxed Set, Books 1-5 (Harry Potte...	J.K. Rowling	4.77	34107	156	âœ¿Katherine Elizabethâœ¿	5.0	NaN
2	3	Harry Potter and the Sorcerer's Stone	J.K. Rowling	4.44	4911929	77741	Lora	5.0	I'm going to keep this brief since there isn't...
3	1	Harry Potter and the Half-Blood Prince (Harry ...	J.K. Rowling	4.54	1810404	28053	Cait (Paper Fury)	5.0	"Read Harry Potter!" they said. "It'll be fun!...
4	2	Harry Potter and the Order of the Phoenix (Har...	J.K. Rowling	4.47	1862749	29308	Diane ÏŸ [Lestrange]	5.0	NaN
5	4	Harry Potter and the Chamber of Secrets (Harry...	J.K. Rowling	4.38	1936698	35055	ZoÃ«	5.0	NaN
6	7	The Harry Potter Collection (Harry Potter, #1-6)	J.K. Rowling	4.73	26702	909	Jen Holman	5.0	I do not own this spiffy box set of Harry Pott...
7	5	Harry Potter and the Prisoner of Azkaban (Harr...	J.K. Rowling	4.53	2000827	37005	Inge	5.0	NaN
8	8053	Charlie Oink (Easy Peasy People)	Roger Hargreaves	4.33	3	0	Ashton	5.0	NaN
9	6	Harry Potter and the Goblet of Fire (Harry Pot...	J.K. Rowling	4.53	1897821	31817	ZoÃ«	5.0	NaN
10	9643	My Country Right or Left: 1940-1943 (The Colle...	George Orwell	4.30	445	24	Rosa	4.0	NaN
11	9644	Why I Write	George Orwell	4.04	5428	479	Matthias	5.0	Part 56 in the "Another autobiographical revie...
12	7209	Bella Tuscany & Under the Tuscan Sun (2 Book Set)	Frances Mayes	3.81	429	21	Catsalive	1.0	NaN
13	3732	Selected Poems of Herman Melville	Herman Melville	3.90	40	4	Keith	3.0	Herman Melvilleâ€™s poetry is an enigma. The m...
14	3404	The Senator and the Socialite: The True Story ...	Lawrence Otis Graham	3.92	154	34	Kenne Jones	4.0	NaN
15	3731	Great Short Works of Herman Melville	Herman Melville	4.05	463	19	James	5.0	In the spring of 1853 after the failure of his...
16	3729	Vance Packard & American Social Criticism	Daniel Horowitz	3.00	1	0	Jay Smith	3.0	NaN
17	7214	Blue Like Jazz: Nonreligious Thoughts on Chris...	Donald Miller	3.92	83897	3598	Seth T.	NaN	I thought of several different ways in which t...
18	3733	Redburn, White-Jacket, Moby-Dick	Herman Melville	4.18	942	68	Ken	5.0	I'm speechless. OK, maybe a few words. At firs...
19	3730	The Hidden Persuaders	Vance Packard	3.91	763	58	Alan	NaN	This slim volume, already more than half a cen...

Table4.1 Data Head

Here we see that our dataset includes numerous columns, but for the purpose of building our recommendation system, we will only use the following columns:

- reviewerName
- title
- reviewerRatings

Note that line 312,075 has an “error bad line” which we will skip, since one row does not have significant impact on the entirety of the 300K+ sample size.

```
# Filters the items that we need
df = df.filter(items=['reviewerName', 'title', 'reviewerRatings'])
df.head(20)
```

	reviewerName	title	reviewerRatings
0	Charles G	Unauthorized Harry Potter Book Seven News: "Ha...	3.0
1	âœ¿Katherine Elizabethâœ¿	Harry Potter Boxed Set, Books 1-5 (Harry Potte...	5.0
2	Lora	Harry Potter and the Sorcerer's Stone	5.0
3	Cait (Paper Fury)	Harry Potter and the Half-Blood Prince (Harry ...	5.0
4	Diane ÏŸ [Lestrange]	Harry Potter and the Order of the Phoenix (Har...	5.0
5	ZoÃ«	Harry Potter and the Chamber of Secrets (Harry...	5.0
6	Jen Holman	The Harry Potter Collection (Harry Potter, #1-6)	5.0
7	Inge	Harry Potter and the Prisoner of Azkaban (Harr...	5.0
8	Ashton	Charlie Oink (Easy Peasy People)	5.0
9	ZoÃ«	Harry Potter and the Goblet of Fire (Harry Pot...	5.0
10	Rosa	My Country Right or Left: 1940-1943 (The Colle...	4.0
11	Matthias	Why I Write	5.0
12	Catsalive	Bella Tuscany & Under the Tuscan Sun (2 Book Set)	1.0
13	Keith	Selected Poems of Herman Melville	3.0
14	Kenne Jones	The Senator and the Socialite: The True Story ...	4.0
15	James	Great Short Works of Herman Melville	5.0
16	Jay Smith	Vance Packard & American Social Criticism	3.0
17	Seth T.	Blue Like Jazz: Nonreligious Thoughts on Chris...	NaN
18	Ken	Redburn, White-Jacket, Moby-Dick	5.0
19	Alan	The Hidden Persuaders	NaN

Table4.2 Modified Data Head(1)

Behavior data is easy to collect because you can keep a log of user activities on your site. Collecting this data is also straightforward because it doesn't need any extra action from the user; they're already using the application. The downside of this approach is that it's harder to analyze the data. For example, filtering the needful logs from the less needful ones can be cumbersome.

Since each user is bound to have different likes or dislikes about a product, their data sets will be distinct. Over time as you 'feed' the engine more data, it gets smarter and smarter with its recommendations so that your email subscribers and customers are more likely to engage, click and buy. Just like how the Amazon's recommendation engine works with the 'Frequently bought together' and 'Recommended for you' tab.

Data Cleaning

Now that our dataset is filtered down to the columns that we will be using for our model, our next step will be data cleaning by performing the following:

- Check for missing data
- Remove all data that has missing values since we cannot make predictions without actual reviewerName, title or reviewerRatings
- Remove all non-ASCII characters as we see a lot of symbols that we cannot interpret
- Reset the index of our DataFrame after data is cleaned so that all rows are indexed sequentially

```
# Checks for missing value for each column
column = ['reviewerName', 'title', 'reviewerRatings']
for columns in column:
    missing = df[columns].isnull().value_counts()
    print(missing)
```

```
False    311532
True        541
Name: reviewerName, dtype: int64
False    312073
Name: title, dtype: int64
False    230898
True      81175
Name: reviewerRatings, dtype: int64
```

```
# Drops missing values for all columns
df = df.dropna(how='any')

# Remove rows with non-ASCII characters in reviewerName and title column
df = df[~df.reviewerName.str.contains(r'[^\x00-\x7F]')]
df = df[~df.title.str.contains(r'[^\x00-\x7F]')]

# Resets the index
df = df.reset_index(drop=True)

# Check and see the cleaned data
df.head(20)
```

	reviewerName	title	reviewerRatings
0	Charles G	Unauthorized Harry Potter Book Seven News: "Ha...	3.0
1	Lora	Harry Potter and the Sorcerer's Stone	5.0
2	Cait (Paper Fury)	Harry Potter and the Half-Blood Prince (Harry ...	5.0
3	Jen Holman	The Harry Potter Collection (Harry Potter, #1-6)	5.0
4	Inge	Harry Potter and the Prisoner of Azkaban (Harr...	5.0
5	Ashton	Charlie Oink (Easy Peasy People)	5.0
6	Rosa	My Country Right or Left: 1940-1943 (The Colle...	4.0
7	Matthias	Why I Write	5.0
8	Catsalive	Bella Tuscany & Under the Tuscan Sun (2 Book Set)	1.0
9	Keith	Selected Poems of Herman Melville	3.0
10	Kenne Jones	The Senator and the Socialite: The True Story ...	4.0
11	James	Great Short Works of Herman Melville	5.0
12	Jay Smith	Vance Packard & American Social Criticism	3.0
13	Ken	Redburn, White-Jacket, Moby-Dick	5.0
14	Mackay	Moon Metro Paris	3.0
15	Anne Hawn Smith	Morality For Beautiful Girls (No. 1 Ladies' De...	5.0
16	Anne Hawn Smith	Morality for Beautiful Girls (No. 1 Ladies' De...	5.0
17	Donna	The Good Husband of Zebra Drive (No. 1 Ladies'...	5.0
18	Kara	Paris to the Moon	5.0
19	Laurel	In the Company of Cheerful Ladies (No. 1 Ladie...	4.0

Fig4.3 Modified Data Head(2)

Data Analysis

Now that our data is cleaned, we have a remaining sample size of 217K+ which is still large enough to make meaningful recommendations. We will also analyze our data briefly, before feeding the data into our model.

```
# Check how many times each books are rated
from collections import Counter
Counter(df['title'].head(20))
```

```
Counter({'Unauthorized Harry Potter Book Seven News: "Half-Blood Prince" Analysis and Speculation': 1,
         "Harry Potter and the Sorcerer's Stone": 1,
         'Harry Potter and the Half-Blood Prince (Harry Potter, #6)': 1,
         'The Harry Potter Collection (Harry Potter, #1-6)': 1,
         'Harry Potter and the Prisoner of Azkaban (Harry Potter, #3)': 1,
         'Charlie Oink (Easy Peasy People)': 1,
         'My Country Right or Left: 1940-1943 (The Collected Essays, Journalism & Letters, Vol. 2)': 1,
         'Why I Write': 1,
         'Bella Tuscany & Under the Tuscan Sun (2 Book Set)': 1,
         'Selected Poems of Herman Melville': 1,
         "The Senator and the Socialite: The True Story of America's First Black Dynasty": 1,
         'Great Short Works of Herman Melville': 1,
         'Vance Packard & American Social Criticism': 1,
         'Redburn, White-Jacket, Moby-Dick': 1,
         'Moon Metro Paris': 1,
         "Morality For Beautiful Girls (No. 1 Ladies' Detective Agency)": 1,
         "Morality for Beautiful Girls (No. 1 Ladies' Detective Agency, #3)": 1,
         "The Good Husband of Zebra Drive (No. 1 Ladies' Detective Agency, #8)": 1,
         'Paris to the Moon': 1,
         "In the Company of Cheerful Ladies (No. 1 Ladies' Detective Agency, #6)": 1})
```

Fig4.1 Times Book Rated

Here we see that there are books that got rated multiple times.

```
# See how many times different reviewers rated the same title
df.groupby("reviewerName")["title"].unique().head(20)
```

```
reviewerName
\tErica                                                [Recuerda Cuerpo...: Poesia Erotica, Ultimate ...
  LunaBel                                              [Don DeLillo's White Noise (Bloom's Modern Cri...
 ( A Bald Mage) Keith                                     [Prism of the Night: A Biography of Anne Rice]
 (shan) Littlebookcove                                 [The Wicker Man, The Third Horror (99 Fear Str...
 - The Book Rack                                       [Arthur C. Clarke: 2001/A Space Odyssey, The C...
 Aesha                                                        [Intimacy: Trusting Oneself and the Other]
 Barb Bailey                                           [301 Country Christmas Quilt Blocks, Do-It-You...
 Caleb                                                 [Christopher Davis's Best Year Yet: Hyperion C...
 Danielle The Book Huntress (Back to the Books)        [Night Embrace (Dark-Hunter, #3), Lucky You, T...
 Debra (WifeEclectic)                                  [Baby Chronicles: My Very Own Story: from pre-...
 Doris Powell                                                          [Psalms for Zero Gravity]
 Dr. Michael Galvin                                    [God Isn't Here: A Young American's Entry into...
 Fernando Brito                                        [Front Row Center 2: Inside the World's Greate...
 Jan  Vajda                                            [John Grisham: A Reader's Checklist and Refere...
 Jordan                                                [The Day It Rained Hearts, Four Valentines in ...
 Julie                                                            [Sacra Pagina: The Gospel of Luke]
 Lady Jayne *~*The Beach Bandida*~*                               [Room Service (Do Not Disturb #6)]
 Linda (Miss Greedybooks)                              [Detroit Is My Own Home Town, Herbalism: Using...
 Marla                                                                      [Sheriff Luke Ludd]
 Martin                                                [Pandora's Box: Pandora's Box, Spawn of Hell, ...
Name: title, dtype: object
```

Fig4.2 Reviewers rated the same title

Here we see that there are not many books that were rated by different users.

```
# See all the different users that rated "Anne of Avonlea" multiple times
same_names = df[df['title'] == 'Anne of Avonlea']["reviewerName"].unique()
for name in same_names:
    print(name)
```

Maureen

Based on our analysis so far:

- First we checked to see if certain books were rated multiple times
- Then we confirmed if the books were rated multiple times by different users
- Next we took a random book called “Anne of Avonlea,” which has 12 separate ratings, and checked to see how many users rated that book
- As a result, we see that only “Maureen” rated the book “Anne of Avonlea,” with an occurrence of 12 ratings

The purpose of our analysis was to confirm if there were two users who rated the same book multiple times, so that we can use theses reviews and see how close they rank amongst each other, using the Euclidean Distance Model.

Since we were not able to manually/experimentally find a book that were rated by the two users above, then our next step will be creating a function for finding the Euclidean Distance between “Maureen” and other users with similarly rated books. Once we have a list of these Euclidean Distances, we can then rank the similarity levels for recommendations.

Note: We have reduced the number of rows down to 20 rows for more web friendly displays. As a result, you may not see the “Anne of Avonlea” book in the output above. However, in order to display the full dataset, we just need to remove the *.head(20)* method in each of the three lines under “Data Cleaning” and “Data Analysis” section.

Data Preparation

In order to fit the data into our model, we need to manipulate our *DataFrame* as following:

{

reviewerName: { title : reviewerRatings }

}

Next, we will transform our *DataFrame* into a dictionary for model testing.

To visualize this, we will use the *sort_index()* method below.

```
# Filters the unique reviewerName for their corresponding title and reviewRatings
df1 = df.set_index(['reviewerName', 'title']).sort_index()
df1.head(20)
```

reviewerName	title	reviewerRatings
\tErica	Recuerda Cuerpo...: Poesia Erotica	3.0
	Ultimate Lesbian Erotica 2005	3.0
LunaBel	Beyond Grief and Nothing: A Reading of Don Delillo	4.0
	Conversations with Don DeLillo	4.0
	Don DeLillo's White Noise (Bloom's Modern Critical Interpretations)	3.0
	Don DeLillo: The Physics of Language	5.0
	Hippolytos	3.0
	Hippolytus	3.0
	Reading Simulacra: Fatal Theories for Postmodernity	3.0
	The Day Room	4.0
	The Enigma of Health: The Art of Healing in a Scientific Age	5.0
	The Hippolytus of Euripides	3.0
(A Bald Mage) Keith	Prism of the Night: A Biography of Anne Rice	4.0
	Prism of the Night: A Biography of Anne Rice	4.0
(shan) Littlebookcove	The Third Horror (99 Fear Street: The House of Evil, #3)	3.0
	The Wicker Man	5.0
	The Wicker Man	5.0
- The Book Rack	Arthur C. Clarke: 2001/A Space Odyssey, The City And The Stars, The Deep Range, A Fall Of Moondust, Rendevous With Rama	4.0
Aesha	Intimacy: Trusting Oneself and the Other	3.0
Barb Bailey	301 Country Christmas Quilt Blocks	3.0

Table4.4 Data Preparation

```
# Converts dataframe to dictionary
d = (df.groupby('reviewerName')['title','reviewerRatings'].apply(
    lambda x: dict(x.values)).to_dict())
# Prints the first 20 items in our dictionary
n = 20
{key:value for key,value in list(d.items())[0:n]}
```

```
{'\tErica': {'Recuerda Cuerpo...: Poesia Erotica': 3.0,
  'Ultimate Lesbian Erotica 2005': 3.0},
 '  LunaBel ': {"Don DeLillo's White Noise (Bloom's Modern...)": 3.0,
  'Hippolytus': 3.0,
  'Hippolytos': 3.0,
  'The Hippolytus of Euripides': 3.0,
  'Conversations with Don DeLillo': 4.0,
  'Beyond Grief and Nothing: A Reading of Don Delillo': 4.0,
  'Don DeLillo: The Physics of Language': 5.0,
...
 ' Marla': {'Sheriff Luke Ludd': 3.0},
 ' Martin': {"Pandora's Box: Pandora's Box": 4.0,
```

'Spawn of Hell': 5.0,
'Idole.': 5.0,
'The Movie': 5.0}}

Note that we may run into an error showing "IOPub data rate exceeded", this issue can be solved here.

Now that we finished cleaning, analyzing and preparing our data, we will build a function to run our model.

Modeling

The model we are using to determine similarities between users is the Euclidean Distance Model, which measures the distance between users, and takes calculates the closest distance for similarity considerations.

Here is the formula that we are going to use:

$dist((x,y),(a,b)) = \sqrt{((x-a)^2 + (y-b)^2)}$

First we will create a "sim_distance" function by implementing the Euclidean Distance formula between two users. We will also add the results by 1 and then divide it by 1 so the Euclidean Score is between 0 to 1.

Next, we will create a "top_matches" function, while also using the "sim_distance" function to determine the list of all other users who are similar to "Maureen". This is also where we are able to find the Euclidean Distance between "Maureen" and other users without manually/experimentally finding two users that rated the same book.

Finally, we will put everything together and create a "get_recommendations" function as our recommendation system. To create our recommendation system, we will use the following approach:

- Firstly, we will calculate the similarity score between all users that are closest to "Maureen" using the "sim_distance" function, this will provide us with a similarity score for each user that is similar to "Maureen"
- Secondly, we will take the similarity score of each user and multiply it by the book rating for the same user, on a book which "Maureen" has yet to rate
- Now, we have a obtained a new value called rated similarity score for each user
- Thirdly, we will add the rated similarity score for all users as our sum of rated similarity score, as well as adding the similarity score for all users as our sum of similarity score
- Fourthly, we will divide the sum of rated similarity score by the sum of similarity score based on ratings by all other users, to determine and recommended books that "Maureen" would be interested in purchasing

We will repeat this process for each book in our “get_recommendations” function for predicting and recommending similar books that “Maureen” is interested in.

```
# Returns a distance-based similarity score for person1 and person2
def sim_distance(prefs,person1,person2):
    si = {}
    for item in prefs[person1]:
        if item in prefs[person2]:
            si[item]=1
    if len(si) == 0:
        return 0
    sum_of_squares = sum([pow(prefs[person1][item] - prefs[person2][item],2)
                          for item in prefs[person1] if item in prefs[person2]])
    return 1/(1+sum_of_squares)

# Checks for similarity to Maureen using Euclidean distance score
def top_matches(prefs, person, n=10, similarity = sim_distance):
    scores = [(similarity(prefs,person,other), other)
              for other in prefs if other!=person]
    scores.sort()
    scores.reverse()
    return scores[0:n]

# Gets recommendation using a weighted average of all other users using
Euclidean distance score
def get_recommendations(prefs, person, n=10, similarity = sim_distance):
    totals = {}
    simSums = {}
    for other in prefs:
        if other == person:
            continue
        sim = similarity(prefs, person, other)
        if sim <= 0:
            continue
        for item in prefs[other]:
            if item not in prefs[person] or prefs[person][item] == 0:
                totals.setdefault(item,0)
```

```
            totals[item] += prefs[other][item] * sim
            simSums.setdefault(item,0)
            simSums[item] += sim
    rankings = [(total/simSums[item], item) for item, total in totals.items()]
    rankings.sort()
    rankings.reverse()
    return rankings[0:n]
```

User Similarities and Recommendations

```
# Top 10 similar users to Maureen
top_matches(d, 'Maureen')
```

```
[(1.0, 'Sebastienne Rimbaud'),
 (1.0, 'Karol'),
 (1.0, 'Jonathan'),
 (1.0, 'Ahmad Sharabiani'),
 (0.5, 'Jeffrey Keeten'),
 (0.5, 'Herb'),
 (0.5, 'Brian'),
 (0.5, 'Brenda'),
 (0.5, 'Bobby Sterr'),
 (0.2, 'Marcy')]
```

```
# Top recommendation for Maureen
get_recommendations(d, 'Maureen')
```

```
[(5.0, 'the story of panda bears'),
 (5.0, 'Your Secrets Sleep With Me'),
 (5.0,
  "You Wouldn't Want to Work on the Great Wall of China!: Def..."),
 (5.0, 'Yoga For Children: A Complete Illustrated Guide to Yoga'),
 (5.0,
  'Writing Security: United States Foreign Policy and the Politi...'),
 (5.0, "Writer's Workshop: Writing Skills and Activities, Grades 4-6"),
 (5.0, 'World of the Thibaults'),
 (5.0, 'Words for a Deaf Daughter and Gala: A Fictional Sequel'),
```

```
 (5.0, 'Word Family Mini-Storybooks: Grades 1-3'),
 (5.0, 'Wonderland Avenue')]
```

After running our model, we see the top 10 users that are similar to Maureen's preferences using the Euclidean Distance. Also, we made a list of recommended books for "Maureen" based on the similarities of all other users to "Maureen", and their respective books that they rated.

Answering the Questions

From the recommendations above, we see that the model was able to recommend books to a consumer that has yet to read/rate, based on similarities of other consumers and their respective book ratings.

With these recommendations, we have created a model that can make predictions and recommend books to consumers.

In addition, we also determined similar preferences of other consumers in comparison to our the original consumer, and vice versa.

Lastly, based on our model's ability make recommendations and determine interests of other consumers, we conclude that our model can effective help a retail book store improve their marketing strategies, and control inventory, which can potentially translate into higher sales and profits.

CHAPTER 5

WHAT ARE THE DIFFERENT TYPES OF RECOMMENDATIONS?

There are basically three important types of recommendation engines:

- Collaborative filtering
- Content-Based Filtering
- Hybrid Recommendation Systems

Collaborative filtering:

This filtering method is usually based on collecting and analyzing information on user's behaviors, their activities or preferences and predicting what they will like based on the similarity with other users. A key advantage of the collaborative filtering approach is that it does not rely on machine analyzable content and thus it is capable of accurately recommending complex items such as movies without requiring an "understanding" of the item itself.Collaborative filtering is based on the assumption that people who agreed in the past will agree in the future, and that they will like similar kinds of items as they liked in the past. For example, if a person A likes item 1, 2, 3 and B like 2,3,4 then they have similar interests and A should like item 4 and B should like item 1.

Further, there are several types of collaborative filtering algorithms:

- **User-User Collaborative filtering:** Here, we try to search for lookalike customers and offer products based on what his/her lookalike has chosen. This algorithm is very effective but takes a lot of time and resources. This type of filtering requires computing every customer pair information which takes time. So, for big base platforms, this algorithm is hard to put in place.
- **Item-Item Collaborative filtering:** It is very similar to the previous algorithm, but instead of finding a customer look alike, we try finding item look alike. Once we have item look alike matrix, we can easily recommend alike items to a customer who has purchased any item from the store. This algorithm requires far fewer resources than user-user collaborative filtering. Hence, for a new

customer, the algorithm takes far lesser time than user-user collaborate as we don't need all similarity scores between customers. Amazon uses this approach in its recommendation engine to show related products which boost sales.

- **Other simpler algorithms:** There are other approaches like market basket analysis, which generally do not have high predictive power than the algorithms described above.

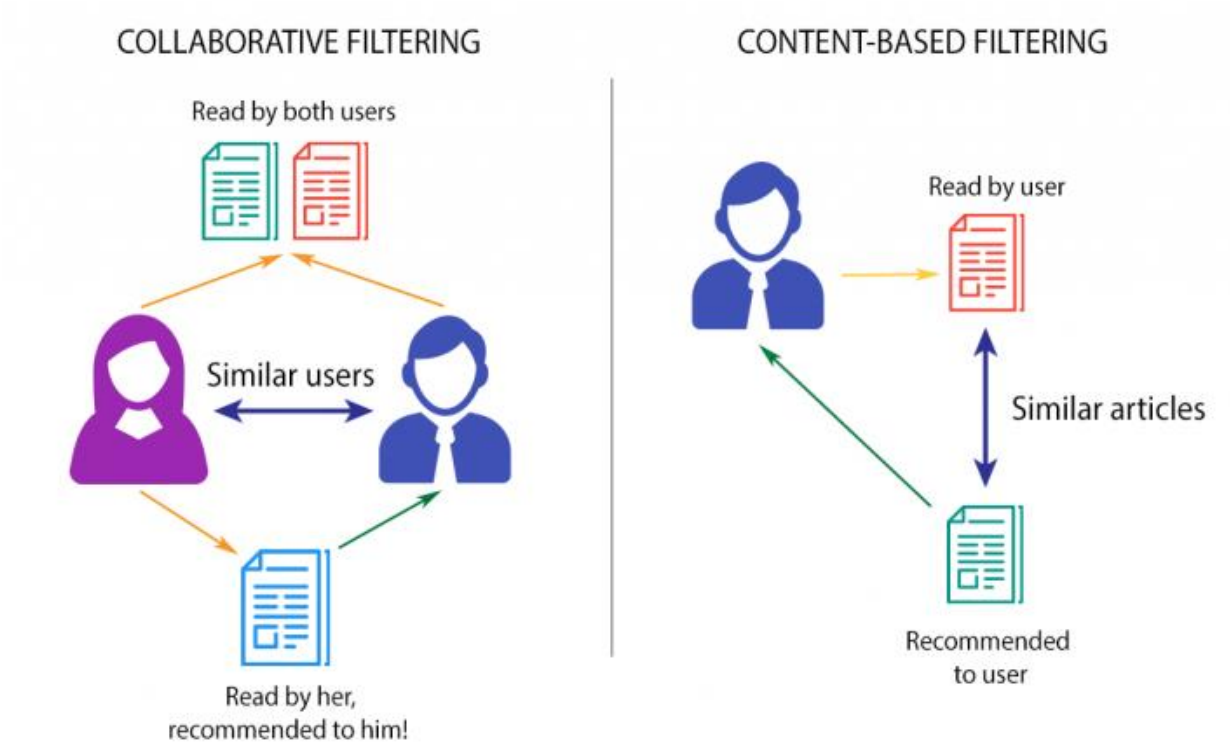

Fig5.1 Types of Filtering

Content-based filtering:

These filtering methods are based on the description of an item and a profile of the user's preferred choices. In a content-based recommendation system, keywords are used to describe the items; besides, a user profile is built to state the type of item this user likes. In other words, the algorithms try to recommend products which are similar to the ones that a user has liked in the past. The idea of content-based filtering is that if you like an item you will also like a 'similar' item. For example, when we are recommending the same kind of item like a movie or song recommendation. This approach has its roots in information retrieval and information filtering research.

A major issue with content-based filtering is whether the system is able to learn user preferences from users actions about one content source and replicate them across other different content types. When the system is limited to recommending the content of the same type as the user is already using, the value from the recommendation system is significantly less when other content types from other services can be recommended. For example, recommending news articles based on browsing of news is useful, but wouldn't it be much more useful when music, videos from different services can be recommended based on the news browsing.

Hybrid Recommendation systems:

Recent research shows that combining collaborative and content-based recommendation can be more effective. Hybrid approaches can be implemented by making content-based and collaborative-based predictions separately and then combining them. Further, by adding content-based capabilities to a collaborative-based approach and vice versa; or by unifying the approaches into one model.

Several studies focused on comparing the performance of the hybrid with the pure collaborative and content-based methods and demonstrate that hybrid methods can provide more accurate recommendations than pure approaches. Such methods can be used to overcome the common problems in recommendation systems such as cold start and the data paucity problem.

Netflix is a good example of the use of hybrid recommender systems. The website makes recommendations by comparing the watching and searching habits of similar users (i.e., collaborative filtering) as well as by offering movies that share characteristics with films that a user has rated highly (content-b ased filtering).

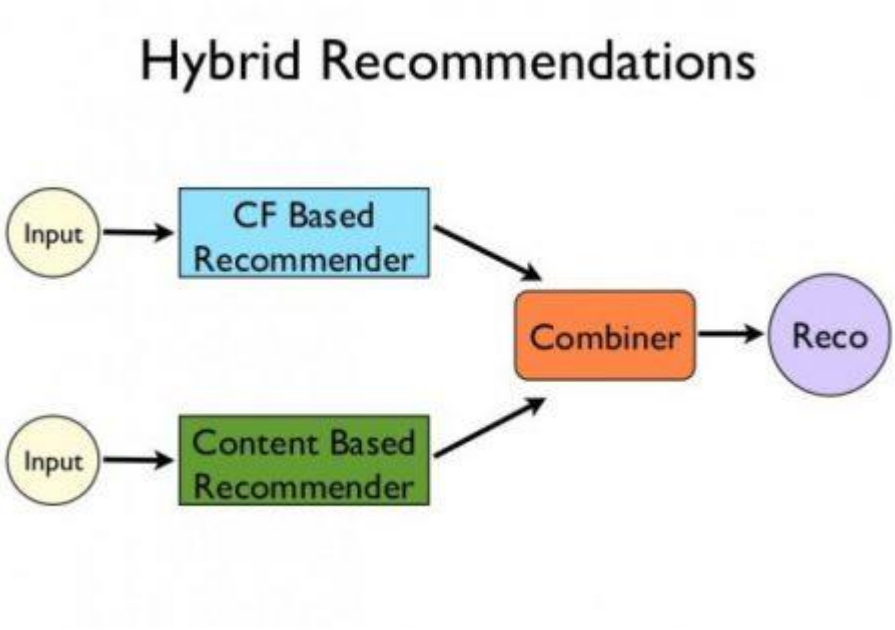

Fig5.2 Hybrid Recommendations

CHAPTER 6
FEASIBILITY STUDY

1. Personalized recommendation→recommend things based on the individual's past behavior.
2. Social recommendation→recommend things based on the past behavior of similar users.
3. Item recommendation→recommend things based on the item itself.
4. A combination of the three approaches above.

CHAPTER 7
SYSTEM REQUIREMENTS

(7.1) Hardware Components:

Hardware include, two Laptop with Intel® Core™ i5-2450M processor, 4GB3 DDR3 SDRAM at 1600MHz, 500GB 5400 RPM SATA Hard Drive.

(7.2) Software Requirement:

Anaconda Navigator: Anaconda Navigator is a desktop graphical user interface (GUI) included in Anaconda® distribution that allows you to launch applications and easily manage conda packages, environments, and channels without using command-line commands. Navigator can search for packages on Anaconda Cloud or in a local Anaconda Repository. It is available for Windows, macOS, and Linux.

Pycharm: **PyCharm** is an integrated development environment (IDE) used in computer programming, specifically for the Python language. It is developed by the Czech company JetBrains. It provides code analysis, a graphical debugger, an integrated unit tester, integration with version control systems (VCSes), and supports web development with Django as well as Data Science with Anaconda.

Python IDE: **Python** code editors are designed for the developers to code and debug program easily. Using these **Python IDE**(Integrated Development Environment), you can manage a large codebase and achieve quick deployment. Developers can use these editors to create desktop or web application.

CHAPTER 8

ADVANTAGES AND DISADVANTAGES

(8.1) Advantages:

1. **Revenue** — With years of research, experiments and execution primarily driven by Amazon, not only is there less of a learning curve for online customers today. Many different algorithms have also been explored, executed, and proven to drive high conversion rate vs. non-personalized product recommendations.

2. **Customer Satisfaction** — Many a time, customers tend to look at their product recommendation from their last browsing. Mainly because they think they will find better opportunities for good products. When they leave the site and come back later; it would help if their browsing data from the previous session was available. This could further help and guide their e-Commerce activities, similar to experienced assistants at Brick and Mortar stores. This type of customer satisfaction leads to customer retention.

3. **Personalization** — We often take recommendations from friends and family because we trust their opinion. They know what we like better than anyone else. This is the sole reason they are good at recommending things and is what recommendation systems try to model. You can use the data accumulated indirectly to improve your website's overall services and ensure that they are suitable according to a user's preferences. In return, the user will be placed in a better mood to purchase your products or services.

4. **Discovery** — For example, the "Genius Recommendations" feature of iTunes, "Frequently Bought Together" of Amazon.com makes surprising recommendations which are similar to what we already like. People generally like to be recommended things which they would like, and when they use a site which can relate to his/her choices extremely perfectly then he/she is bound to visit that site again.

5. **Provide Reports** — Is an integral part of a personalization system. Giving the client accurate and up to the minute, reporting allows him to make solid decisions about his

site and the direction of a campaign. Based on these reports clients can generate offers for slow moving products in order to create a drive in sales.

(8.2) Disadvantages:

1. Lack of Data

Perhaps the biggest issue facing recommender systems is that they need a lot of data to effectively make recommendations. It's no coincidence that the companies most identified with having excellent recommendations are those with a lot of consumer user data: Google, Amazon, Netflix, Last.fm. As illustrated in the slide below from Strands' presentation at Recked, a good recommender system firstly needs item data (from a catalog or other form), then it must capture and analyze user data (behavioral events), and then the magic algorithm does its work. The more item and user data a recommender system has to work with, the stronger the chances of getting good recommendations. But it can be a chicken and egg problem – to get good recommendations, you need a lot of users, so you can get a lot of data for the recommendations.

2. Changing Data

This issue was pointed out in ReadWriteWeb's comments by Paul Edmunds, CEO of 'intelligent recommendations' company Clicktorch. Paul commented that systems are usually "biased towards the old and have difficulty showing new".

An example of this was blogged by David Reinke of StyleHop, a resource and community for fashion enthusiasts. David noted that "past behavior [of users] is not a good tool because **the trends are always changing**" (emphasis ours). Clearly an algorithmic approach will find it difficult if not impossible to keep up with fashion trends. Most fashion-challenged people – I fall into that category – rely on trusted fashion-conscious friends and family to recommend new clothes to them.

David Reinke went on to say that "item recommendations don't work because there are simply too many product attributes in fashion and each attribute (think fit, price, color, style, fabric, brand, etc) has a different level of importance at different times for the same consumer." He did point out though that social recommenders may be able to 'solve' this problem.

3. Changing User Preferences

Again suggested by Paul Edmunds, the issue here is that while today I have a particular intention when browsing e.g. Amazon – tomorrow I might have a different intention. A classic

example is that one day I will be browsing Amazon for new books for myself, but the next day I'll be on Amazon searching for a birthday present for my sister (actually I got her a gift card, but that's beside the point).

On the topic of user preferences, recommender systems may also incorrectly label users – a la this classic Wall St Journal story from 2002, If TiVo Thinks You Are Gay, Here's How to Set It Straight.

4. Unpredictable Items

In our post on the Netflix Prize, about the $1 Million prize offered by Netflix for a third party to deliver a collaborative filtering algorithm that will improve Netflix's own recommendations algorithm by 10%, we noted that there was an issue with eccentric movies. The type of movie that people either love or hate, such as Napoleon Dynamite. These type of items are difficult to make recommendations on, because the user reaction to them tends to be diverse and unpredictable.

Music is full of these items. Would you have guessed that this author is a fan of both Metallica and The Carpenters? I doubt Last.fm would make that recommendation.

CHAPTER 9

Problem analysis and solution

9.1.Problem analysis

4.1.1. The challenge

As we mentioned previously, our objective is to build a hybrid recommender system that is able to achieve higher prediction accuracy than the ordinary single recommender systems. Furthermore,we are aiming at integrating it in the most seamless way possible the recommender system into Liferay portal. In order to improve the performance of the recommender system, we are planning to combine both the content features and rating information of the problem domain. Building a recommender presents two main challenges which are: (1) making the right design decisions for the system to be built and (2) finding the best way to represent the data so that it can be efficiently used by the recommender to provide higher prediction accuracy than single recommender systems. Another major challenge that we have to face and which regards the integration strategy, is finding the best strategy possible so that after the integration both the requirements of the host system and those of the recommender are still met. Such a strategy must in addition, enable the hosted application to benefit from the capabilities of the host application.

In order to take up the above mentioned challenges we are going to study the guidelines that drive the design of recommender systems. Doing so will enable us to make the right choices for the design of our system. In addition to studying the guidelines, we are going also to analyze the advantages and disadvantages of the integrations strategies for Liferay portal. From this analysis we are going to decide which strategy to use in order to take up the challenge regarding that aspect.

Several works in the literature provided many guidelines of many aspects of building a recommender system but they describe general principles that should guide the design of the recommender engine once the initial technological decisions have been made. In this subsection we are going to address the following question:

How can we make sensible choices when initially designing the system?

Designing a recommender system means making choices that can be categorized into the following domains:

- Algorithms: which recommendation methods to use?
- Architecture: how will the system be deployed, will it be centralized or distributed?
- User profile: what is the user model? Is profile adaptation needed?

These choices are constrained for a large part by the environment of the recommender. The environment of the recommender can be studied through three dimensions:

- **Users:** who are the users, what are their goals?

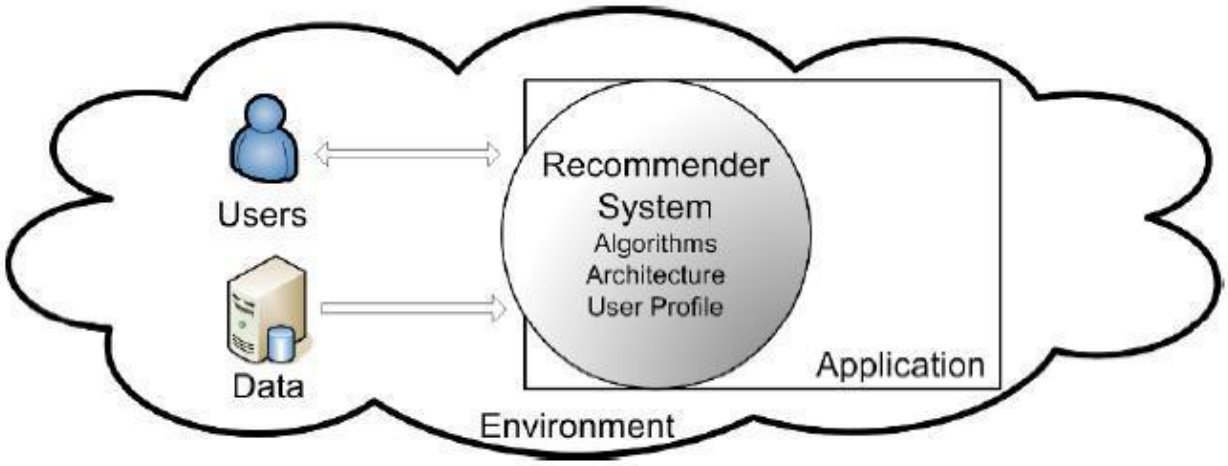

FIGURE 24 THE RECOMMENDER SYSTEM IN ITS ENVIRONMENT

- **Data:** what are the characteristics of the data on which recommendations are based?
- **Application:** what is the overall application the recommender is part of?

Application model

Though a recommender system is itself a complex piece of software, it is by nature part of a larger system. A recommender is one of the features of an overall application. It may be a minor feature or a main selling point; the application maybe pre-existing or built together with the recommender, but in any case the design of a recommender system has to be integrated within the design of the application hosting it. This section studies the main factors regarding the host application that should influence the recommender design along two main lines: the role of the recommender and the influence of the application implementation. In order to understand the role of the recommender system in the portal the following questions had to be answered:

- ✓ What is the purpose of the recommender of engine?
- ✓ What is the type of recommendations that it should provide?

- ✓ Will the recommendations integrate with other content navigation features? And how?
- ✓ What metrics will be used to measure the recommender engine's performance?

Purpose of the recommender engine: the recommender may be a major service provided by the application hosting it (in our case the portal). But it can be of a lesser importance than the other services provided by the application. Furthermore, a recommender within the context of a business may give easy access to previously hard to find items as it is the case in many e-commerce web sites. Customers return to the services that best match their needs. Ultimately the purpose of a recommender system may be is the increase of the system efficiency by allowing the user to more directly get the content she is looking for. This in return can lower the amount of data to be exchanged, thus lowering the costs of running a system. In clear, the different objectives or purposes that a recommender may have are: be a major service or not, enable easy access to previously hard to find items, secure users' loyalty and increase the system's efficiency.

Type of recommendations: a recommender can provide several sorts of recommendations, from a single item (or a simple list of items) to a sequence (e.g. in a travel recommender system). Single item or simple list recommenders do not take into account how the choice of an item by the user at a given point of time may influence the choice of next items. The choice of the recommendation type may be driven by the need or presence of a logical order in the recommendations. For example, in a travel recommender system, a trip can be seen as a sequence of travel steps (such as visiting a museum, going to the beach, etc.), which can be connected through various logical features, such as geography, culture, history, leisure, etc. in order to provide a good travel experience. Recommendations of sequences of items may be particularly useful when users are new to a domain and need a path in the selection of diverse items, helping them to go through their personal development goals: the logical order of the recommendations help them progressing in their learning curve by providing the next most appropriate step(s).

Integration with the navigation content: how the recommendations will integrate with other content navigation features? In most cases, users will be offered other means to browse content in addition to getting recommendations. A good integration of these different navigation methods can greatly enhance the user experience. Users may request recommendations completely separately from content browsing. This can be a good choice if recommendations are to be highlighted as a main feature of the application. Such recommendations may also appear on the home page of a web site or home screen of an application. It can also be beneficial to have recommendations dependent on the current

interaction context. The typical case is to recommend items that are similar to those the user is currently browsing. In that case, the recommender system must be able to provide recommendations tailored to the context, e.g. the current category when browsing news article (if the current articles' category is sports then the recommender must recommend items in that category).

It is also important to consider whether the use of the recommender system is optional or a mandatory part of the interaction model. This has strong implications on the expected reliability of the system: failure to complete a major task on a website because the only way of completing that task was using a recommender system which offered inaccurate recommendations could be a source of major user dissatisfaction. However within a system design where the recommender sat in parallel to more traditional navigation methods, the impact of the same recommender may be many times less severe.

Performance metrics: The choice of the metrics not only will allow evaluating the recommender once it is built but also will influence the choice of the algorithms to implement it. The key metrics we considered are as it follows:

- *Correctness metrics such as accuracy, precision and recall:* they are technical criteria which can be used to evaluate recommender system. Accuracy is the margin of error the recommender make while recommending items. Precision refers to the proportion of top recommendations that are good recommendation. Recall represents is the proportion of good recommendations that appear in top recommendations.
- *Transparency and explainability:* it shows how important is it that users understand how the recommendations have been determined. A good level of transparency can be more difficult to achieve with some families of algorithms.
- *Serendipity:* should users be (hopefully pleasantly) surprised by some of the recommendations or is it desirable to allow obvious recommendations?
- *Risk taking*: related to the previous criterion, should the recommendations be made only for items that the user has a high probability of liking? More risky items can be recommended if the goal is to allow the user to discover content they would not be aware of without the help of the system. But the designer of a recommender must be careful of the risks they take in exploring new items. The cost of the users' dissatisfaction must be taken into consideration before taking such risks.
- *Response speed / performance*: in many cases, the reactivity of the application is a major concern and can be sometimes more important than the accuracy of the results. Knowing how many recommendations are needed per time unit allows to better choose algorithms or decide if recommendations should be pre-computed.

- *Reliability*: What is the criticality of the recommender output in the context of the given application? For instance the design of a recommender for an ecommerce website would not be approached in the same way as a solution for an organ donor matching system in a hospital.

- *Robustness* to attacks: in particular if the recommender system has a commercial role (for

instance if it recommends products for purchase), it may be subject to attacks to skew the results. After we have understood the first line (the role of the recommender within the context of the portal) which influences the design of the recommender and made the necessary choices, we have to understand the influence of the implementation of the hosting application on the design of the recommender system.

In addition to the features seen by the users, some aspects of the host application implementation also have a large influence on how the recommender can be designed.

Single or multiple devices: the same application may be accessed from a single or multiple devices (e.g. a news recommender system on mobile, PC…).

It must be studied whether the recommendations should depend on the user context. But in term of implementation, it also raises additional questions:

Should the collected preferences be merged or should they remain separate to enable contextual recommendations? Where should the preferences be stored? If the preferences are stored on a server, are they transmitted to the server in real-time (implying a constant connection) or in batches? Answering such questions is important even if access from multiple devices is not initially planned, as it is becoming the norm that web applications are derived into mobile versions.

Single or multiple users: conversely, the same device may be used by several users. In addition, users that interact with the host application may be either registered or anonymous and may interact frequently or occasionally. This impacts the architecture of the recommender (requires different identification means, e.g. login vs. cookies), algorithm choice (e.g. session profile in case of anonymous occasional users vs. persistent profile in case of registered users), and algorithm parameters (e.g. the degree of adaptability of the system to the user – i.e. the rapidity of profile adaptation: long-term vs. short-term – should depend on the frequency of use).

User model

Fully understanding the user is a fundamental component to the success of any recommender system. In this section we discuss the properties on the user's side which may have an impact on the design and choices we as the designers of the recommender faced. Identifying those properties boils down to understanding who the users are, and what expectations and goals lie behind the users motivations to use the system the recommender supports.
Understanding who the users are revolves around three main concerns: understanding their key identifying characteristics, their skills levels and their prior experience with similar systems. The most important concern in our case is:

Identifying users' characteristics: Gathering a picture of the different user groups through both demographic information such as age and gender, job area, nationalities, spoken languages, and deep qualitative insights from user research are an important jumping off point in the development of recommender user models. Understanding these factors allows the development team to start to build a relationship with the users and get an appreciation of their needs. Development of user group clusters may allow (1) the building of simple recommenders based on demographics. This is commonly used in targeted advertising solutions to cluster customers into segments; (2) define stereotypes of users: stereotyping techniques allow the definition of a set of differentiating characteristics for a group of users; when a new user is introduced into the system, they can be assigned to a predefined stereotype, based on their personal data, which allows the activation of a set of default preferences that may be further refined over time thanks to user profile adaptation methods . Personalization solutions exploiting user characteristics can be used in combination with more sophisticated techniques to provide a first simple step in a hybrid filtering process, or to bootstrap content-based filtering algorithm by using stereotype profiles.
In order to understand what the users' motivation, expectation and goals are, the designer of the recommender engine needs to identify the users' tasks and understand if the application can support completion. For instance the Amazon web site is offering its users the possibility to buy items and get recommendations of items to buy. From the user's viewpoint, their motivation for using the service is to complete one of the two goals of either buying an item for themselves, or buying an item for someone else. The Amazon recommender however, does not differentiate those two goals and therefore provides inaccurate recommendation results in one of the use cases. Identifying and understanding user motivation can result in fundamental recommender and user experience improvements. User insights captured within the user model allow designers to consider the possible range of tasks the future application needs to support.

In most cases there will be many motivations for the use of a system, and a designer must consider ways of finding the nature of the goal, either explicitly or implicitly. An *explicit* goal may be either defined by the application (Amazon could have added an option asking the user if they were buying the item for themselves or someone else) or expressed by the user (for example through a set of queries). An *implicit* goal may be either predefined by the application itself (such as in personalized news pushing systems where the system provides stories to the user in the belief that the user goal is to feel more informed about particular world events), or can be inferred from the user's interactions. In contrast it is quite possible that a clear task goal is not discernible (e.g. because the application is dedicated to enjoyment).

In such cases it can be difficult to build a good recommender as user satisfaction may be more strongly related to external factors such as user mood outside of the system's control. In this case, spread, quantity or serendipity may be the most desirable qualities of the recommender.

Data model

The last point the designer should study carefully are the characteristics of the items the system will exploit and manipulate. Indeed, item descriptions generally preexist before the recommender system, and the designer of the recommender system has little possibility to influence or change them. The designer of the recommender system needs to identify the main characteristics of the data that may influence the design and the results of the recommender system. The following set of questions will help the designer in this task:

- ✓ What kind of data will the recommender manipulate? Is it structured, semi-structured or unstructured?
- ✓ How are the quality and quantity of the metadata? Are they high, medium or low?
- ✓ Are the metadata keyword-based or semantic-based?
- ✓ What is the volume of items? A lot, few?
- ✓ How is the diversity of the items? Heterogeneous, homogeneous?
- ✓ Are the items changing a lot or stable?
- ✓ Are the user ratings implicit or explicit?

- ✓ Are the ratings binary or multi-level?

User items ratings

Another important consideration is that of taking into account user ratings or not. If there is no user rating related to items in the system, this excludes the whole family of collaborative filtering methods. User ratings about items can be either *explicit* (for example on Amazon, users can indicate how much they enjoyed a book or a CD), and may be expressed on different scales (e.g. binary, multi-level). If this is not directly available, but thought to be useful in the personalized application, it is possible to compute *implicit feedback indicators* (for example, the fact of purchasing an item can be considered as an implicit indicator of user interest), which can be used either

CHAPTER 10

Solution

10.1. Design choices for the recommender engine and integration strategy

As we have seen in section 4.1.2, developing a recommender engine requires that we follow certain guidelines. These guidelines are nothing but a set of questions the developer of the recommender system must answer in order to properly identify the constraints and challenges they have to face up to.

In this section, we discuss in one hand, the decisions we took for the implementation of the recommender system and the reasons why they were taken. On the other hand we discuss the integration strategy we opted for and explain as well why we went for it.

Designing a recommender system means making choices that can be categorized into the following domains:

- Algorithms: which recommendation methods to use?
- Architecture: how will the system be deployed, will it be centralized or distributed?
- User profile: what is the user model? Is profile adaptation needed?

These choices are constrained for a large part by the environment of the recommender. The environment of the recommender can be studied through three dimensions:

- **Users:** who are the users, what are their goals?
- **Data:** what are the characteristics of the data on which recommendations are based?
- **Application:** what is the overall application the recommender is part of?

Algorithms: Which recommendation methods to use?

As we mentioned previously, the approach we are going to use for our intended recommender engine is a hybrid approach which combines content-based filtering and collaborative filtering. The reason why we opted for this approach resides in the analysis of the advantages and pitfalls of the different flavors of recommender systems in the literature (section 2.2.3). The analysis showed that content based filtering had some problems such as the *limited content analysis* problem, the *over-specialization* problem and the *new user* problem. It also showed that collaborative filtering had problems like the *cold start* problem, and scalability problems such as the *synonymy* problem and the *new item* problem or first rater problem. The approach we propose combines content features with user ratings, making it possible to

overcome the limits of both techniques. The *limited content analysis* problem in content based filtering which is nothing but the result of situations where items' features are not expressive enough to model the users' interest is overcome by the collaborative filter which uses the users' ratings to model their interest. The *over-specialization* problem and the *new user* problems are also overcome using user ratings. The *cold start* problem and the *new rater* problems in collaborative filtering which are nothing but the result of data scarcity (scarcity in user ratings), making it hard for the collaborative filter to provide accurate recommendations, are overcome using the content features which introduce enough additional data for the hybrid recommender to use.

In order to decide which architecture and what the user model will be, the developer of the recommender system must take into account the environment of the recommender system. This environment can be studied under the light of three models which are: the application model, the user model and the data model. Section 4.1.2 studied these models already. Here we are going to discuss and explain the choices we made for each factor in these models.

Application model

Purpose of the recommender engine: the main goal of the recommender system we intend to build is to enable its users to have easy access to previously hard to find items, and also to increase the efficiency of the system.

Type of recommendations: the type of recommendations the recommender engine is going to generate is single items recommendations. The recommender will not recommend aggregated list of items of various types but instead will recommend list of a single type of items. Within the context of the Global world labor corp.'s portal, a list of recommended job vacancies may be an example of single items recommendations.

Integration with navigation content: The integration with navigation content will be tight. Recommendations will be integrated into the traditional content.

Performance metric: correctness, response speed, reliability, transparency and serendipity are the metrics we are going to use to evaluate the recommender system.

Single or multiple devices: For the current version of the recommender engine we decided to go for mainly fixed devices only. However given that Liferay offers the possibility to expose interfaces for mobile devices, subsequent versions of the recommender may include interfaces for mobile devices.

Single or multiple users: Given that generally the users of the GLWC portal request recommendations for themselves we decided that the recommender engine will generate recommendations for single users.

Application infrastructure: the application will be browser based. Given that Liferay portal's presentation layer is browser based, the recommender system will adopt the same paradigm.

Screen real estate: as we mentioned previously the recommendations generated by the recommender will be integrated into traditional content of the portal. There is therefore a necessity to specify how much space these recommendations will occupy on the screen. We decided to allocate forty five percent of the screen space to display recommendations.

CHAPTER 9

EXPERIMENTAL RESULTS

This paper used Root Mean Square Error (RMSE) statistical accuracy metrics to evaluate the online book recommendation system. RMSE measures the variation between the predicted value and actual user rating. The predicted rating is calculated as follows:

$$= \Sigma \quad , (\quad , \quad * \quad ,)$$

$$, \qquad \frac{}{\Sigma \quad , (| \quad , |)}$$

Hence, smaller the value of RMSE, better the recommendation system. The computed value of RMSE for this system is 1.504.

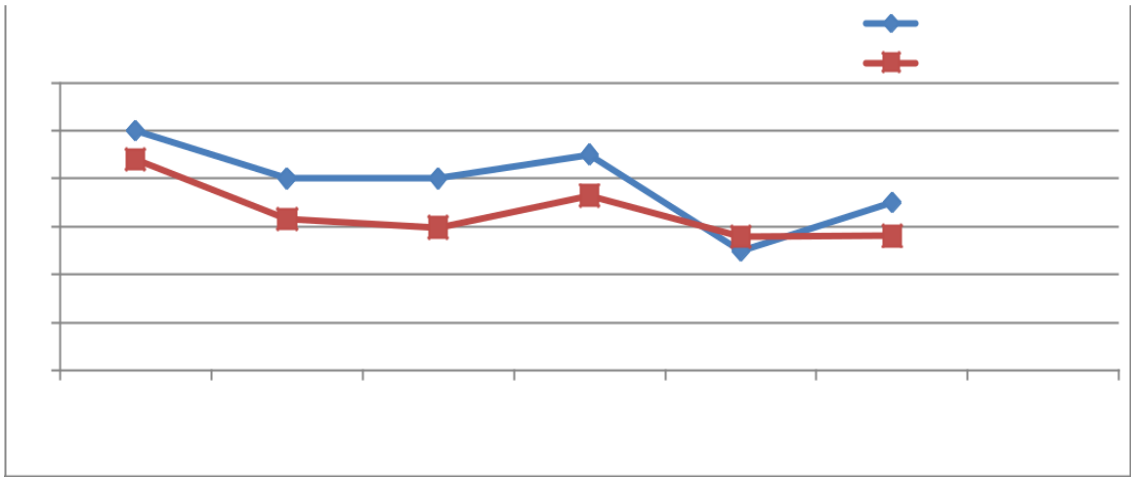

Fig.9.1 Comparison of Ratings and Predictions for user-item pairs

CONCLUSION

Even with the adaptation of a fitting algorithm for recommendation, the RS faces an obstacle because of large quantity of data that needs to be handled. According to the experimental results, the proposed algorithm with compact dataset was more accurate than existing algorithms with full datasets. In addition, JS uses the number of common users as a basis for measuring similarity, rather than the absolute ratings, as used by most existing algorithms, which gives a more accurate result.

FUTURE WORK

The recommendation system proposed here takes the number of users who have rated the books into account, without factoring in the absolute rating. Due to this, a recommendation might arise from a book that a user has given low rating to, in which case a book might be recommended from a genre that the user dislikes.

This recommendation system relies on the ratings given by users. So, trust is a major issue, like whether the feedback and rating given by the user is genuine or not. This recommendation system does not solve the trust issue.

Therefore future research should focus on resolving both these issues.

REFERENCES

[1] Elgohary, A., Nomir, H., Sabek, I., Samir, M., Badawy, M. and Yousri, N.A. (2010). Wiki-rec: A semantic-based recommendation system using Wikipedia as ontology. In 10th International Conference on Intelligent Systems Design and Applications (ISDA).

[2] Oku, K., Nakajima, S., Miyazaki, J. and Uemura, S. (2006). Context-aware SVM for context-dependent information recommendation. In MDM'06 proceedings of International Conference on Mobile Data Management.

[3] Ghani, R. and Fano, A. (2002). Building recommender systems using a knowledge base of product semantics. In proceedings of 2nd International Conference on Adaptive Hypermedia and Adaptive Web Based Systems.

[4] Miyahara, K. and Pazzani, M.J. (2000) Collaborative filtering with the simple Bayesian classifier. In proceedings of Pacific Rim International Conference on Artificial Intelligence.

[5] Asanov D. (2011) Algorithms and Methods in Recommender Systems. Berlin Institute of Technology Berlin, Germany

[6] Lakshmi, S.S. and Lakshmi, T.A. (2014). Recommendation Systems: Issues and challenges. (IJCSIT) International Journal of Computer Science and Information Technologies, Vol. 5 (4), 2014, 5771-5772

[7] Okon, E.U., Eke, B.O. and Asagba, P.O. (2018). An improved online book recommender system using collaborative filtering algorithm. International Journal of Computer Applications(0975-8887) Volume 179-No.46, June 2018

[8] Kurmashov, N., Konstantin, L., Nussipbekov, A. (2015). Online book recommendation System. Proceedings of Twelve International Conference on Electronics Computer and Computation (ICECC)

[9] Mathew, P., Kuriakose, B. And Hegde, V. (2016). Book Recommendation System through content based and collaborative filtering method. Proceedings of International Conference on Data Mining and Advanced Computing (SAPIENCE)

[10] Parvitikar, S. and Dr. Joshi, B. (2015). Online book recommendation system by using collaborative filtering and association mining. Proceedings of IEEE International Conference on Computational Intelligence and Computing Research (ICCIC)

[11] Ayub, M., Ghazanfar, M.A., Maqsood, M. and Saleem, A. (2018). A Jaccard base similarity measure to improve performance of CF based recommendation system. Proceedings of International Conference on Information Networking (ICOIN)

[12] Gogna, A., Majumdar, A. (2015). A Comprehensive Recommender System Model: Improving Accuracy for Both Warm and Cold Start Users. IEEE Access Vol. 3, 2803-2813, 2015

[13] Chatti, M.A., Dakova, S., Thus, H. and Schroeder, U. (2013). Tag-Based Collaborative Filtering Recommendation in Personal Learning Environments. IEEE Transactions on Learning Technologies, Vol. 6, No. 4, October-December 2013

[14] Choi, S.H., Jeong, Y.S. and Jeong, M.K. (2010). A Hybrid Recommendation Method with Reduced Data for Large-Scale Application. IEEE Transactions on Systems, Man and Cybernetics-Part C: Applications and Reviews, Vol. 40, No.5, September 2010.

[15] Liu, Q., Chen, E., Xiong, H., Ding, C.H.Q. and Chen, J. (2012). Enhancing Collaborative Filtering by User Interest Expansion via Personalised Ranking. IEEE Transactions on Systems, Man and Cybernetics-Part B: Cybernetics, Vol. 42, No.1, February 2012.

[16] Feng, J., Fengs, X., Zhang, N. and Peng, J. (2018). An improved collaborative filtering method based on similarity. PLOS ONE 13 (9): e0204003, September 2018.

[17] Aggarwal, C.C. (2016). Recommendation System: The Textbook. XXI, 29 p. ISBN 978-3-319-29657-9

[18] NHK K. ISMAIL*,"Estimation Of Reliability Of D Flip-Flops Using Mc Analysis", Journal of VLSI Circuits And Systems 1 (01), 10-12,2019.

[19] MN BORHAN,"Design Of The High Speed And Reliable Source Coupled Logic Multiplexer",Journal of VLSI Circuits And Systems 1 (01), 18-22,2019