

# **Project Report**

*On*

Book Recommendation System using Content Based (Tag Based) Filtering



*Submitted in partial fulfilment of the requirements of the degree*

*Of*

**Bachelor of Technology**

*In*

**Computer Engineering**

*Under the guidance of*

**Prof. Jalaj Sharma**

(Assistant Professor)

**By:**

DEEKSHA ARORA                      50385

MAANSI BHATNAGAR                50387

RAKSHITA JOSHI                      50388

***Department of Computer Engineering, College of Technology,***

***Govind Ballabh Pant University of Agriculture and Technology,***

***Pantnagar-263145, India***

### ACKNOWLEDGEMENT

This project is not the work of solely our team but it's a result of combined efforts put in by many people. We would like to take this opportunity to thank all the people who helped us to carry out this project.

First of all, we express our gratitude to our project guide, **Prof. Jalaj Sharma** without whose help the completion of this project would not have been possible. It was his mentorship that encouraged us to expedite our project process and complete it in time. His precious suggestions and constructive guidance has been indispensable in the completion of this project work.

We would also like to thank **Prof. S.D. Samantaray, Prof. B. K. Singh, Prof. Rajeev Singh, Prof. P.K. Mishra, Prof. Chetan Singh Negi , Prof. Sunita Jalal , Prof. Sukhwinder Singh, Prof. Pragya Kamal and Prof. Deepak Kumar** for providing all the help and infrastructure needed for the successful completion of this project. They have supported us in this endeavor and appreciated us in our efforts during our project.

Last but not the least, we would also like to thank our friends who directly and indirectly supported us during the project work and provided a helpful environment for our project.

Deeksha Arora (50385)

Maansi Bhatnagar (50387)

Rakshita Joshi (50388)

Dated: 23 July 2020

---

## BOOK RECOMMENDATION SYSTEM

---



### CERTIFICATE

This is to certify that the project work entitled “**Book Recommendation System using Content Based (Tag Based) Filtering**” which is submitted by:

Name	Id. No.
DEEKSHA ARORA	50385
MAANSI BHATNAGAR	50387
RAKSHITA JOSHI	50388

is a record of a students work carried by them in partial fulfillment of requirements for the award of degree of Bachelor of Technology in the Department of Computer Engineering, College of Technology, G.B. Pant University of Agriculture and Technology, Pantnagar.

PANTNAGAR

Prof. Jalaj Sharma

Date: 23 JULY 2020

(Project Guide)

## **BOOK RECOMMENDATION SYSTEM**

---

### **APPROVAL**

This project report entitled “**Book Recommendation System using Content Based (Tag Based) Filtering**” is hereby approved a creditable study of an engineering subject, as a prerequisite to degree for which it has been submitted.

#### **Name of Committee Members:**

1. Prof. Jalaj Sharma .....  
(Project Guide)
2. Prof. B.K Singh .....  
(Associate Professor)
3. Prof. Sunita Jalal .....  
(Assistant Professor)
4. Prof. Pragya Kamal .....  
(Assistant Professor, TEQIP)

**Signature of Head of Department**

**DEPARTMENT OF COMPUTER ENGINEERING**

**COLLEGE OF TECHNOLOGY**

**G.B. Pant University of Agriculture & Technology**

**Pantnagar – 263145, INDIA**

---

### **ABSTRACT**

The vast amount of data available on the Internet has led to the development of recommendation systems. Recommendation systems are tools in e-commerce websites which helps user to find the appropriate products. With the rapid development of internet technologies the number of online book selling websites has increased which enhanced the competition among them. . Recommendation system is one of the stronger tools to increase profit and retaining buyer.

The objective of this project is the implementation of a book recommender system, using the data made available by Goodreads, a website where users can register and rate the books they have read as well as share their ratings and opinions with other readers.

The approach chosen is to generate a system that recommends books using the information inherent in users' ratings. So, rather than predicting the ratings that each user would give to all the books that he hasn't read yet, we have chosen to generate a system that gives relevant recommendations to each user based on a certain measure of similarity between the books he has already read and rated and the books he hasn't read but other similar users have.

This report provides a detailed summary of the project “Book Recommendation System” and includes a description of the topic, system architecture, and provides a detailed description of the work done: snapshots of the implementations, various approaches, and tools used so far. The report also includes the project schedule and deliverables.

## TABLE OF CONTENTS

<b>1. Project Overview.....</b>	<b>10</b>
1.1. Introduction.....	11
1.2. Recommendation System.....	11
1.3. Classification of Recommendation System.....	11
1.3.1. Collaborative Filtering Method.....	12
1.3.2. Content Based Filtering Method.....	14
1.4. Evaluation of Recommendation System.....	14
<b>2. Review of Literature.....</b>	<b>17</b>
<b>3. Problem Specification.....</b>	<b>20</b>
<b>4. Requirement Specification.....</b>	<b>23</b>
4.1. Overview.....	24
4.2. Software Requirement.....	24
4.3. Hardware Used.....	24
<b>5. System Design.....</b>	<b>26</b>
5.1. Python and Associated Package.....	27
<b>6. Methodology.....</b>	<b>29</b>
6.1. Content Based Filtering (Tag Based) Algorithm.....	30
<b>7. Implementation.....</b>	<b>32</b>
7.1. Import the Desired Libraries.....	33
7.2. Loading and Reading of the Data.....	33
7.3. Processing of the Data to the Desired form.....	34
7.4. Content Based Filtering (Tag Based).....	36
<b>8. Testing and Results.....</b>	<b>40</b>
8.1. Testing.....	41

---

## BOOK RECOMMENDATION SYSTEM

---

8.2. Results.....	42
<b>9. Conclusion.....</b>	<b>46</b>
<b>10. References.....</b>	<b>47</b>

---

### **LIST OF FIGURES**

1. Classification of Recommendation System.....	11
2. Pearson correlation.....	31
3. Code Snippet.....	33
4. Content Based Filtering (Tag Based) – RESULTS.....	42
5. Content Based Filtering (Tag Based) - COMPARISION.....	43



**CHAPTER 1**  
**PROJECT OVERVIEW**

## PROJECT OVERVIEW

### 1.1 INTRODUCTION

During the last few decades, with the rise of Youtube, Amazon, Netflix and many other such web services, recommendation systems have taken more and more place in our lives. From e-commerce (suggesting buyer articles that could interest them) to online advertisement (suggesting users the right content matching their preferences), recommendation systems are today unavoidable in our daily online journey.

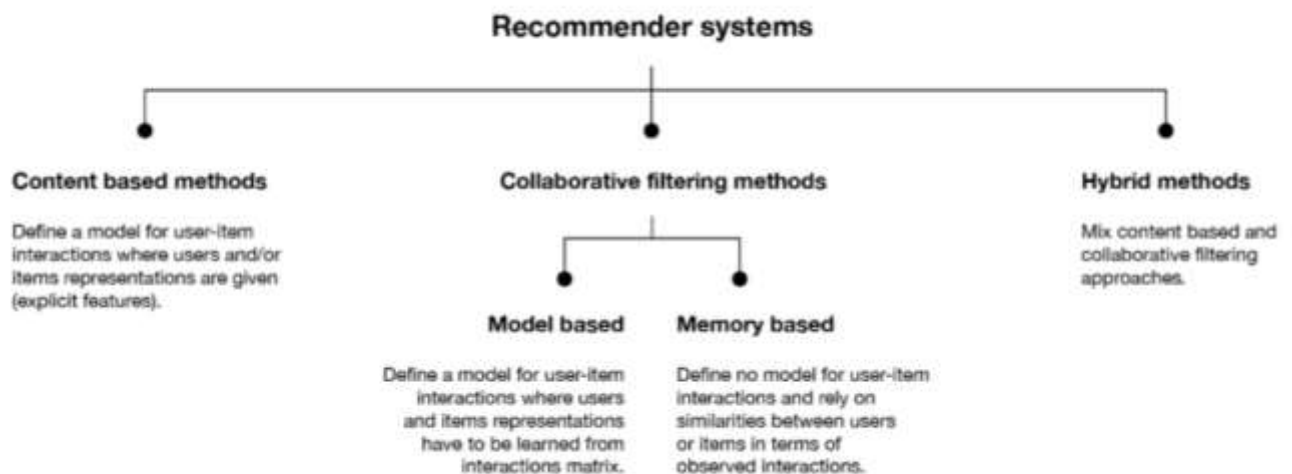
Recommendation systems are really critical in some industries. An efficient system can generate a huge amount of income and can also be a way to stand out significantly from competitors.

The goal of this project is to study recommendation engines and identify the shortcomings of traditional recommendation engines and to develop a web based recommendation engine by making use of a user based collaborative filtering (CF) engine and combining context based results along with it. The System would benefit those users who have to scroll through pages of results to find relevant content.

### 1.2 Recommendation Systems:

Recommendation system is an information filtering technique, which provides users with information, which he/she may be interested in.

### 1.3 Classification of Recommendation Systems



### 1.3.1 Collaborative Filtering Methods

Collaborative methods for recommender systems are methods that are based solely on the past interactions recorded between users and items in order to produce new recommendations. These interactions are stored in the so-called “user-item interactions matrix”. Then, the main idea that rules collaborative methods is that these past user-item interactions are sufficient to detect similar users and/or similar items and make predictions based on these estimated proximities.

The class of collaborative filtering algorithms is divided into two sub-categories that are generally called **memory based** and **model based** approaches. Memory based approaches directly works with values of recorded interactions, assuming no model, and are essentially based on nearest neighbors search (for example, find the closest users from a user of interest and suggest the most popular items among these neighbors). Model based approaches assume an underlying “generative” model that explains the user-item interactions and try to discover it in order to make new predictions.[4]

#### Advantages of Collaborative Approach

The main advantage of collaborative approaches is that they require no information about users or items and, so, they can be used in many situations. Moreover, the more users interact with items the more new recommendations become accurate: for a fixed set of users and items, new interactions recorded over time bring new information and make the system more and more effective.

#### Problems with Collaborative Approach

However, as it only consider past interactions to make recommendations, collaborative filtering suffer from the “cold start problem”: it is impossible to recommend anything to new users or to recommend a new item to any users and many users or items have too few interactions to be efficiently handled.

#### Solution

This drawback can be addressed in different way: recommending random items to new users or new items to random users (random strategy), recommending popular items to new users or new items to most active users (maximum expectation strategy), recommending a set of various items to new users or a new item to a set of various users (exploratory strategy) or, finally, using a non collaborative method for the early life of the user or the item. In the following sections, we will mainly present three classical collaborative filtering approaches: two memory based methods (user-user and item-item) and one model based approach (matrix factorization).

#### Memory Based Collaborative Approach

---

## BOOK RECOMMENDATION SYSTEM

---

The main characteristic of user-user and item-item approach is that they use information only from the user-item interaction matrix and they assume no model to produce new recommendations.

- **User-User**

In order to make a new recommendation to a user, user-user method roughly tries to identify users with the most similar “interactions profile” (nearest neighbors) in order to suggest items that are the most popular among these neighbors (and that are “new” to our user). This method is said to be “user-centred” as it represent users based on their interactions with items and evaluate distance between users.

- **Item-Item**

To make a new recommendation to a user, the idea of item-item method is to find items similar to the ones the user already “positively” interacted with. Two items are considered to be similar if most of the users that have interacted with both of them did it in a similar way. This method is said to be “item-centred” as it represent items based on interactions users had with them and evaluate distance between those items.

### Model Based Collaborative Approach

Model based collaborative approaches only rely on user-item interaction information and assume a latent model supposed to explain these interactions. For example, matrix factorization algorithms consist of decomposing the huge and sparse user-item interaction matrix into a product of two smaller and dense matrices: a user-factor matrix (containing users representation) that multiplies a factor-item matrix (containing items representation).

### 1.3.2 Content Based Filtering Methods

Unlike collaborative methods that only rely on the user-item interactions, content based approach uses additional information about users and/or items. If we consider the example of a movie recommendation system, the additional information can be, for example, the age, the sex or any other personal information of users or categories like, the main actors, the duration or other characteristics of the movie. The idea of content based methods is to build a model, based on the available “features”, that explains the observed user-item interactions. Still considering users and movies, we will try, for example, to model the fact that young women tend to rate better some movies, that young men tend to rate better some other movies and so on. If we manage to get such model, then, making new predictions for a user is pretty easy: we just need to look at the profile (age, sex, ...) of this user and, based on this information we can determine relevant movies to suggest.

Content based methods suffer far less from the cold start problem than collaborative approaches: new users or items can be described by their characteristics (content) and so relevant suggestions

---

can be done for these new entities. Only new users or items with previously unseen features will logically suffer from this drawback, but once the system is old enough, this has few to no chance to happen.

### Concept of Content-Based Methods

In content based methods, the recommendation problem is casted into either a classification problem (predict if a user “likes” or not an item) or into a regression problem (predict the rating given by a user to an item). In both cases, we are going to set a model that will be based on the user and/or item features at our disposal (the “content” of our “content-based” method).

### 1.4 Evaluation of Recommendation System

As for any machine learning algorithm, we need to evaluate the performance of our recommendation system in order to decide which algorithm fits the best. Evaluation methods for recommendation system can mainly be divided in two sets:

- evaluation based on well defined metrics
- evaluation mainly based on human judgment and satisfaction estimation.

#### Metrics based evaluation

If our recommender system is based on a model that outputs numeric values such as ratings predictions or matching probabilities, we can assess the quality of these outputs in a very classical manner using an error measurement metric such as, for example, mean square error (MSE). In this case, the model is trained only on a part of the available interactions and is tested on the remaining ones.

Still if our recommender system is based on a model that predicts numeric values, we can also binarize these values with a classical threshold approach (values above the threshold are positive and values below are negative) and evaluate the model in a more “classification way”. Indeed, as the dataset of user-item past interactions is also binary (or can be binarized by threshold), we can then evaluate the accuracy (as well as the precision) of the binarized outputs of the model on a test dataset of interactions not used for training.

Finally, if we now consider a recommender system not based on numeric values and that only returns a list of recommendations (such as user-user or item-item that are based on a K.N.N. approach), we can still define a precision metric by estimating the proportion of recommended items that really suit our user. To estimate this precision, we cannot take into account recommended items that our user has not interacted with and we should only consider items from the test dataset for which we have user feedback.

---

### Human Based Evaluation

When designing a recommendation system, we can be interested not only to obtain a model that produces recommendations we are very sure about but we also expect some other good properties such as diversity and explainability of recommendations.

As mentioned in the collaborative section, we absolutely want to avoid having a user being stuck in what we called earlier an information confinement area. The notion of “serendipity” is often used to express the tendency a model has or not to create such a confinement area (diversity of recommendations). Serendipity, which can be estimated by computing the distance between recommended items, should not be too low as it would create confinement areas, but should also not be too high as it would mean that we do not take enough into account our users’ interests when making recommendations (exploration vs exploitation). Thus, in order to bring diversity in the suggested choices, we want to recommend items that both suit our user very well and that are not too similar from each other. For example, instead of recommending a user “Star Wars” 1, 2 and 3, it seems better to recommend “Star Wars 1”, “Star Trek into Darkness” and “Indiana Jones and the Raiders of the Lost Ark”: the two later may be seen by our system as having less chance to interest our user but recommending 3 items that look too similar is not a good option.

Explainability is another key point of the success of recommendation algorithms. Indeed, it has been proven that if users do not understand why they have been recommended a specific item, they tend to lose confidence in the recommendation system. So, if we design a model that is clearly explainable, we can add, when making recommendations, a little sentence stating why an item has been recommended (“people who liked this item also liked this one”, “you liked this item, you may be interested by this one”, ...).

Finally, on top of the fact that diversity and explainability can be intrinsically difficult to evaluate, we can notice that it is also pretty difficult to assess the quality of a recommendation that does not belong to the testing dataset: how to know if a new recommendation is relevant before actually recommending it to our user? For all these reasons, it can sometimes be tempting to test the model in “real conditions”. As the goal of the recommendation system is to generate an action (watch a movie, buy a product, read an article etc...), we can indeed evaluate its ability to generate the expected action. For example, the system can be put in production, following an A/B testing approach, or can be tested only on a sample of users. Such processes require, however, having a certain level of confidence in the model.

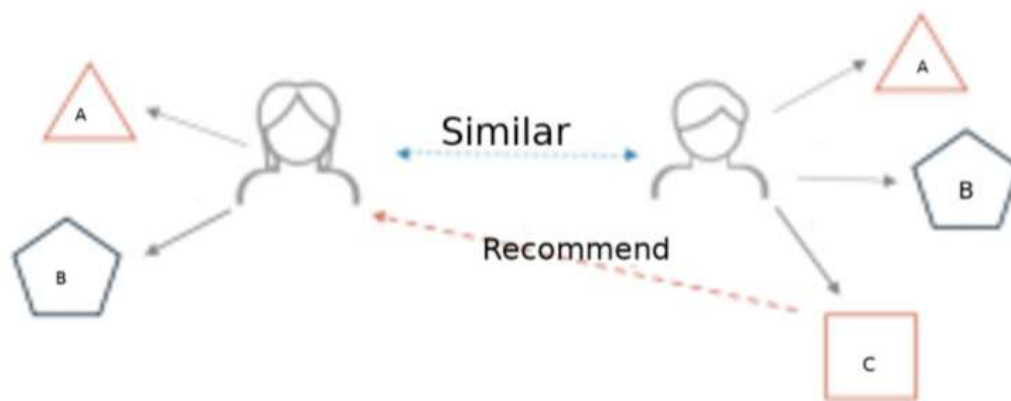
---

**CHAPTER 2**  
**REVIEW OF LITERATURE**

---

### REVIEW OF LITERATURE

In any data analysis model for forecasting, data processing is very important for gaining useful insights from the derived data knowledge. We studied a few papers for devising a model that can process the data into useful insights and correct forecasted output. Following is the pictorial representation of the task we are accomplishing through our project.



Now, as per the topic of our study book recommendation plays a very pivotal role for helping the users to select the book of his choice from the millions of books available out there. This is further dependent on many external independent variables like choice, rating, age group to which person belongs, time the reader can devote to one book and price of the book. Therefore, the accurate book recommendation system is required to help the reader get through all these problems and make the task to choose the book much easier.

Now to devise such a forecasting model, we need an efficient and advanced system that can incorporate all the above factors to predict the book which should be recommended to the reader. At first, a linear regression moving average model was proposed but that gives equal importance to all the independent factors, which results in skewed output. So, a weighted linear regression model was adopted that gives more importance to recent past data and unequal weighted importance to all other factors.



**CHAPTER 3**  
**PROBLEM SPECIFICATION**

---

### **PROBLEM SPECIFICATION**

In the present scenario, many readers have difficulty in selecting the book to read as there are millions of books out there to select from. The factors which affect the reader's choice of book include his age, his interest, the time he could devote to one book, the ratings which he preferred and various other factors, making it difficult for them to choose a suitable book to read. In order to eradicate this problem and make the decision making process easier for the reader, we decided to devise a book recommendation system model, through which the reader can find out the book which suits his taste and will make the process of choosing the book much faster and easier. Besides this, the choice made would also be accurate. We sorted books for the users based on the following factors:

#### **Ratings**

While the choice and selection of any book might be dependent on the content of the book, but the demand highly depend on the ratings i.e. the reviews or the points provided to the book for some particular content. This data is obtained from a site, from where we download all our datasets used in the project. More the rating audience gives for a particular book, the book becomes more valuable and thus favorable to reader. While in the other scenario, if the audience provides low rating for some book then that book will be less preferable to the reader and thus not favorable at all.

#### **Authors**

In any reading world, the book preferred by the user is highly dependent on the author he preferred. Here for our study, we use author's name as one of the factor that can affect the choice and preference of any chosen user. While there are other factors as well but this is very significant. The choice of author helps us to sort the books very easily for the user.

#### **Reviews**

The main source for any prediction is the old reviews of past readers. So here also, we take reviews of certain past readers to help us distinguish between the preferred book for the reader. These reviews are processed and transformed into useful data. From this processed data, we further derive insights for predicting book. This factor constitutes the main part of our sorting.

#### **Applied Learning**

There are different approaches to develop this recommendation system, but we used collaborative filtering. Collaborative filtering is a technique that can filter out items that a user might like on the basis of reactions by similar users. It works by searching a large group of people and finding a smaller set of users with taste similar to a particular user. It looks at the

---

## BOOK RECOMMENDATION SYSTEM

---

items they like and combines them to create a ranked list of suggestions. There are many ways to decide which users are similar and combine their choices to create a list of recommendations. Collaborative filtering is a family of algorithms where there are multiple ways to find similar users or items and multiple ways to calculate rating based on ratings of similar users. One important thing to keep in mind is that in an approach based purely on collaborative filtering, the similarity is not calculated using factors like the age of users, genre of the book, or any other data about users or items. It is calculated only on the basis of the rating (explicit or implicit) a user gives to an item. For example, two users can be considered similar if they give the same ratings to ten books despite there being a big difference in their age.

**CHAPTER 4**  
**REQUIREMENT SPECIFICATION**

---

## REQUIREMENT ANALYSIS

### 4.1 OVERVIEW

The goal of this project is to study recommendation engines and identify the shortcomings of traditional recommendation engines and to develop a web based recommendation system by making use of user based collaborative filtering (CF) engine and combining context based results along with it.

### 4.2 SOFTWARE REQUIREMENT

As the project is developed in python, we have used Anaconda for Python 3.7 and Jupyter Notebook.

#### **Anaconda**

It is a free and open source distribution of the Python and R programming languages for data science and machine learning related applications (large-scale data processing, predictive analytics, scientific computing), that aims to simplify package management and deployment. Package versions are managed by the package management system Anaconda. The Anaconda distribution is used by over 6 million users, and it includes more than 250 popular data science packages suitable for Windows, Linux, and MacOS.

#### **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. Project Jupyter exists to develop opensource software, open-standards, and services for interactive computing across dozens of programming languages.

#### **Software Interfaces**

1. Microsoft Word 2016
2. Database - My SQL 5.6
3. Operating System: Windows10
4. Anaconda 3

### 4.3 HARDWARE USED:

#### **Hardware Requirements:**

1. i5 Processor Based Computer
-

## BOOK RECOMMENDATION SYSTEM

---

2. 4 GB-Ram(Atleast)

3. 100 GB Hard Disk(Atleast)



**CHAPTER 5**  
**SYSTEM DESIGN**

### SYSTEM DESIGN

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

#### **5.1 PYTHON AND ASSOCIATED PACKAGE**

Python was the language of choice for this project. This was an easy decision for the multiple reasons:

1. Python as a language has an enormous community behind it. Any problems that might be encountered can be easily solved by using Stack Overflow.
2. Python has an abundance of powerful tools ready for scientific computing. Packages such as Numpy, Pandas, and SciPy are freely available and well documented. These packages can dramatically reduce and simplify the code needed to write a given program. This makes iteration quick.

The various modules of python used are given as following:

##### **1. NumPy:**

It is the fundamental package for scientific and numerical computing with Python. It defines the numerical array and matrix types and basic operations on them. It contains a powerful N-dimensional array object, sophisticated (broadcasting) functions, tools for integrating C/C++ and Fortran code and useful linear algebra, Fourier transform, and random number capabilities. Besides its obvious scientific uses, NumPy can also be used as an efficient multi-dimensional container of generic data where arbitrary data-types can be defined. This allows NumPy to seamlessly and speedily integrate with a wide variety of databases.

##### **2. Pandas:**

Pandas is a python software package. It provides us with many Series and DataFrames and allows to easily organize, explore, represent, and manipulate data. Smart alignment and indexing featured in Pandas offers a perfect organization and data labeling. Pandas has some special features that allows to handle missing data or value with a proper measure.

##### **3. SciPy:**

Scipy is an open-source python library that is used for both scientific and technical computation. It is used for computation and image manipulation. Scipy contains different modules. These modules are suitable for optimization, integration, linear algebra, and statistics, as well. It makes

---



the best use of Numpy arrays for general data structures. In fact, Numpy is an integrated part of Scipy.

### **4. Matplotlib:**

Matplotlib is a Python library that uses Python Script to write 2-dimensional graphs and plots. Often mathematical or scientific applications require more than single axes in a representation. This library helps us to build multiple plots at a time. Matplotlib can also be used to manipulate different characteristics of figures as well.

### **5. Scikit Learn:**

Scikit-learn is an open source Python library that implements a range of machine learning, pre-processing, cross-validation and visualization algorithms using a unified interface. It is built on the top of NumPy, SciPy, and matplotlib. It provides simple and efficient tools for data mining and data analysis. It features various classification, regression and clustering algorithms including support vector machines, random forests, gradient boosting, k-means, etc.

## **CHAPTER 6**

## **METHODOLOGY**

### METHODOLOGY

#### 6.1 CONTENT-BASED FILTERING (TAG BASED) ALGORITHM

##### 1. Loading the Data

2. Encode Data: For our example, we will use the term frequency–inverse document frequency (TF-IDF) encoding scheme. The advantage of **TF-IDF encoding** is that it will weigh a term (a tag for a book in our example) according to the importance of the term within the document: The more frequently the term appears, the larger its weight will be. At the same time, it weighs the item inversely to the frequency of this term across the entire dataset: It will emphasize terms that are relatively rare occurrences in the general dataset but of importance to the specific content at hand. That means that words such as ‘is’, ‘are’, ‘by’ or ‘a’ which are likely to show up in every movie description but aren’t useful for our user-recommendation, will be weighed less than words that are more unique to the content that we are recommending.

The formula used to calculate TF-IDF weight for term  $i$  in document  $j$  is:

$$w[i,j] = tf[i,j] * \log(N/df[i])$$

$tf$  is the term frequency,  $df$  is the document frequency and  $N$  stands for the total number of documents in the dataset.

A vector-encoded document will look like this when encoded:

```
array([ 1., 0.46036753, 0.16328608, ..., 0.29024403, 0.36014058, 0.23019143])
```

Each element in the vector represents a TF-IDF weight associated with a term in a document.

3. Recommending Content: We are going to use a simple similarity-based method called **cosine similarity**. Vectors have direction and magnitude. Because of this, we can calculate the angle between two vectors. A popular measure in data science is the cosine of this angle computed as follows:

$$\cos(x,y) = \text{dot}(x,y)/|x||y|$$

This measure will equal 1 when the vectors are parallel (they point in the same direction) and 0 when the vectors are orthogonal. Vectors that point in the same direction are more similar than vectors that are orthogonal.

**CHAPTER 7**  
**IMPLEMENTATION**

### IMPLEMENTATION

In our project we have used Sequential Technique of code and implementation as well as code execution. The Steps followed for the proper execution of code and generation of desired output are as follows:

- 7.1 Importing the desired libraries
- 7.2 Loading and Reading of the Data
- 7.3 Processing of the data to desired form
- 7.4 Content-Based Filtering (Tag-Based)

#### 7.1 IMPORTING THE DESIRED LIBRARIES

```
In [1]: import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.feature_extraction.text import TfidfVectorizer, CountVectorizer
from sklearn.metrics.pairwise import linear_kernel, cosine_similarity
from scipy.sparse import csr_matrix
```

Fig 7.1

#### 7.2 LOADING AND READING OF THE DATA

```
In [2]: books = pd.read_csv('books.csv')
ratings = pd.read_csv('ratings.csv')
tags = pd.read_csv('tags.csv')
bookTags = pd.read_csv('book_tags.csv')
toRead = pd.read_csv('to_read.csv')
```

Fig 7.2

The data used for this project consist of 5 different datasets, which are loaded and presented above.

The 5 datasets include the following datasets -

1. **Books** contains all the information about the rated books, including author, title, book ID, publication year, average rating, etc.
  2. **Ratings** includes all the rates given by our selected group of users to the books they have chosen to rate.
-

## BOOK RECOMMENDATION SYSTEM

---

3. **Tags** collects all the tags included in 'bookTags' and explains their meaning.
4. **Book Tags** contains all the tags associated to each and every book included in the analysis.
5. **to Read** indicates all the books that have been flagged as 'to read' by the readers included in the analysis.

### 7.3 PROCESSING OF THE DATA TO THE DESIRED FORM

```
In [47]: # We can now drop the duplicates records for the combination user/book with more than 1 rating
ratings = ratings.drop_duplicates(subset = ['book_id', 'user_id'], keep = 'first', inplace = False) \
    .sort_values('count', ascending = False)

In [48]: ratings = ratings.drop(['rating', 'count'], axis = 1)
```

Fig 7.3

We can now drop the duplicates records for the combination user/book with more than 1 rating as in Fig 7.3

```
In [42]: # We create now a table grouped by user and book pair, calculating mean and number of rating that a user has made for the
# same book

ratings['count'] = ratings['rating']

userRatesPerBook = ratings.groupby(['user_id', 'book_id'], as_index = False) \
    .agg({'rating' : 'mean', 'count' : 'count'}) \
    .rename(columns = {'rating': 'mean'}) \
    .sort_values('count', ascending = False)

ratings = ratings.drop(['count'], axis = 1)

userRatesPerBook['mean'] = userRatesPerBook['mean'].round(0)
userRatesPerBook.head()
```

Out[42]:

	user_id	book_id	mean	count
59787	3204	8946	5.0	5
99001	5091	8946	5.0	4
972	42	8946	5.0	4
42643	2308	8946	5.0	4
665336	34548	7420	5.0	4

Fig 7.4

We now create a table grouped by user and book pair, calculating mean and number of rating that a user has made for the same book as shown in Fig 7.4. With the purpose of making more relevant recommendations, let's have a look at what is the ratings distribution per users: in other words, let's see if there are users who have rated very few books in Fig 7.5

## BOOK RECOMMENDATION SYSTEM

---

```
In [53]: ratesPerUser = ratings[['user_id', 'rating']].groupby(['user_id'], as_index = False) \
        .count() \
        .rename(columns = {'rating' : 'totalRatings'}) \
        .sort_values('totalRatings', ascending = False)

ratesPerUser.head(10)
```

Out[53]:

	user_id	totalRatings
12873	12874	200
30943	30944	200
52035	52036	199
28157	28158	199
12380	12381	199
45553	45554	197
6629	6630	197
19728	19729	196
24142	24143	196
15603	15604	196

Fig 7.5

This histogram below Fig 7.6 shows how many users have rated only 1 to 15 books. Considered that we are using only a subset (10k) of all the books listed in Goodreads, it will be pretty difficult to give accurate recommendation to users with that few ratings.

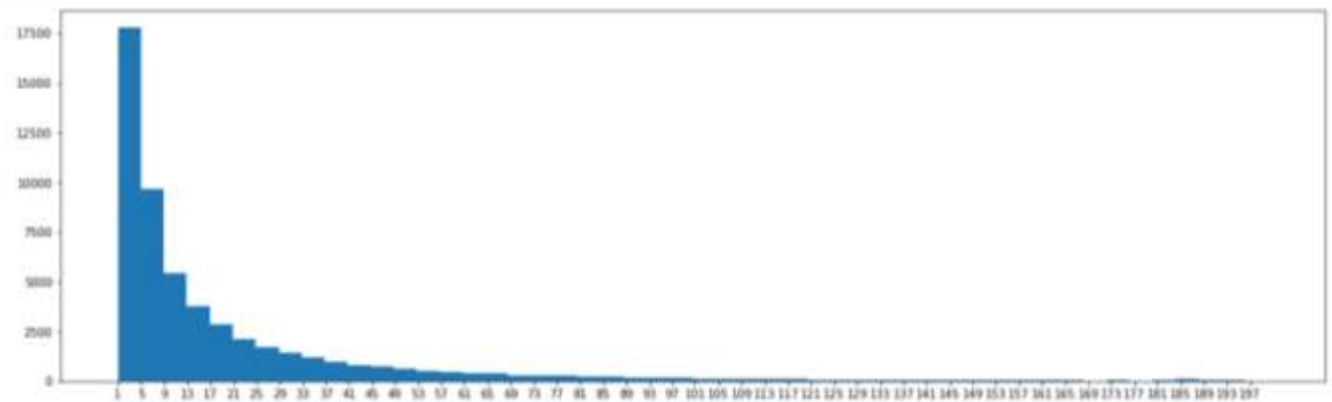


Fig 7.6

Considering also the fact that this project has rather an illustrative purpose, I have decided to include only user that have rated at least 20 of the 10K books included in the analysis. Then I had a look at the same thing but about the books: how many books have very few ratings? Looking at the results, I have decided - for the same reason as above, that is to avoid 'obscure' recommendations of books that only few users have read - to remove books with less than 30 ratings received.

---

## BOOK RECOMMENDATION SYSTEM

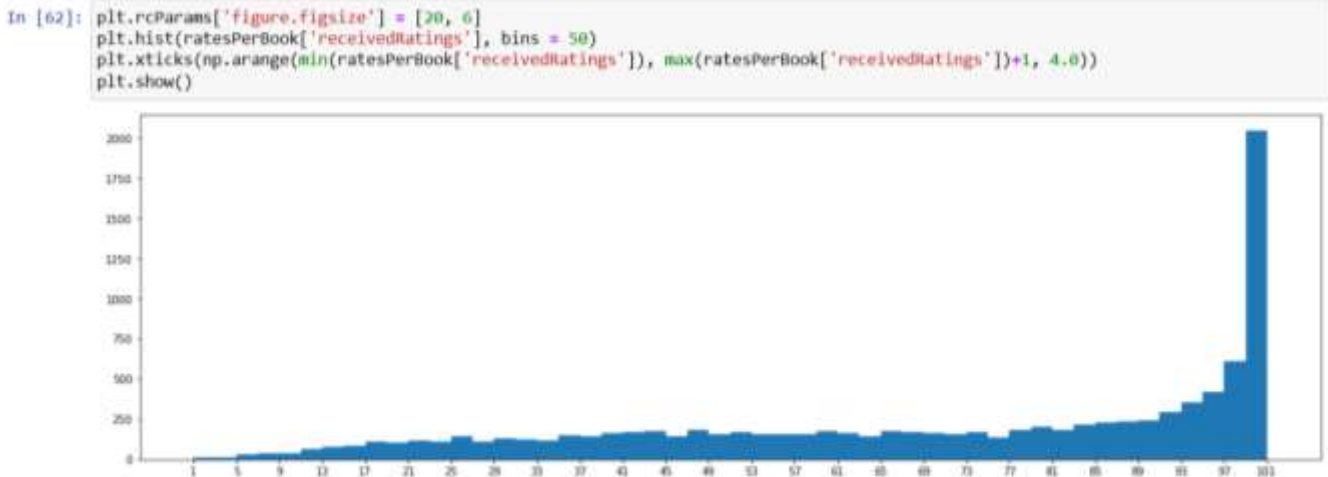


Fig 7.7

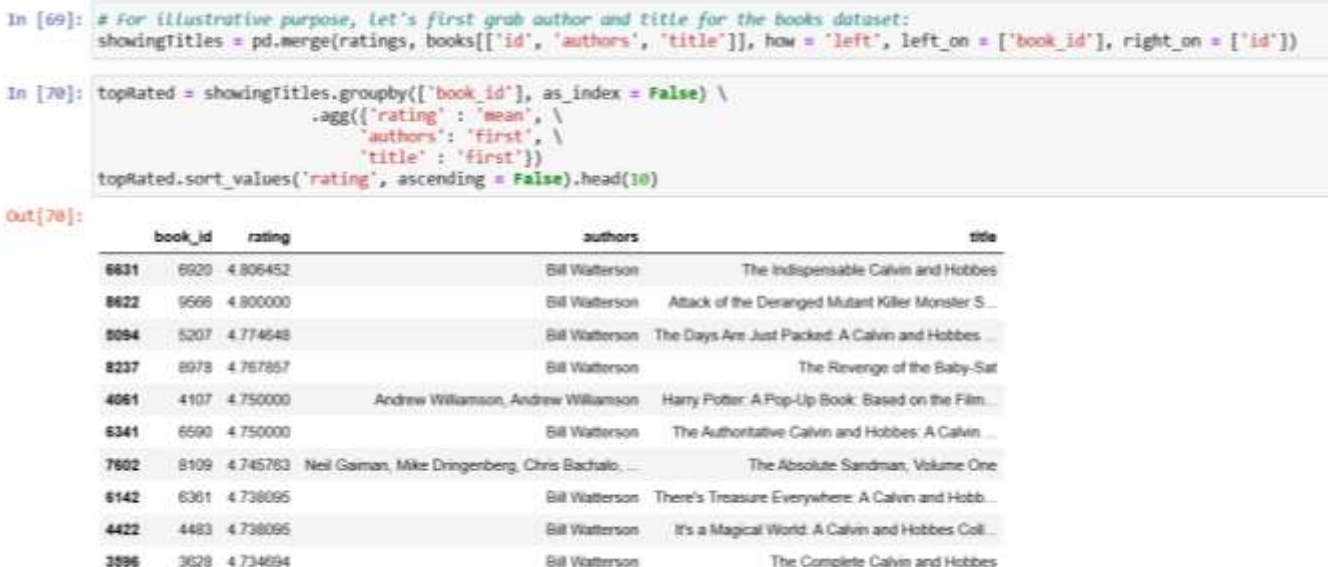


Fig 7.8

To start off, I have tried to calculate a first type of Top-N recommendation in Fig7.8: the top rated books. After having removed the books and users above, the result doesn't change much compared to the what shown right after the data loading: Bill Watterson and his Calvin and Hobbes is still dominating the chart.

### 7.4 CONTENT BASED FILTERING (TAG BASED)

As long as the first Top-N recommendation created doesn't seem at all to be useful, we have build a "Content-Based" recommendation, leveraging the knowledge we have from the tags dataset using tags and other information from the book metadata.

However, as shown below, the tags dataset contains format-free definition, which include a vast variety of tags. Most of them are not saying much of the book itself and its context or



## BOOK RECOMMENDATION SYSTEM

---

characteristics (e.g. 'to-read', 'favourite', 'book-I-own', 'made-me-cry', etc). For this reason we have decide to include only the tags that are representative of the book 'genre' according to Goodreads itself: the tags used in fact are scraped from their genre section and contain a vast variety of tags. The others have been filtered out, So we have used the “genres.csv” files, which was generated by scraping Goodreads website, and it is shown in the other notebook.

```
In [73]: genres = pd.read_csv('genres.csv')
genres.head(10)
```

Out[73]:

	Unnamed: 0	tag_name
0	0	art
1	1	biography
2	2	business
3	3	children-s
4	4	christian
5	5	classics
6	6	comics
7	7	cookbooks
8	8	ebooks
9	9	fantasy

Fig 7.9

Now we used 'CountVectorizer' function, that will generate a matrix in which the columns represent all the tag-words that I have included in the analysis and the rows represent the books.

From that matrix (the 'tagMatrix'), we have calculated a similarity score between each book pair, choosing the 'cosine' as a metric of distance.

```
In [90]: countVec = CountVectorizer(analyzer = 'word', ngram_range = (1, 2), min_df = 0, stop_words = 'english')
```

```
In [91]: tagMatrix = countVec.fit_transform(stringedTags['all_tags'])
```

Fig 7.10

**CHAPTER 8**  
**TESTING AND RESULTS**

### 8.1 TESTING

An essential part of our project was testing. The system has been tested by keeping a small set of data from the “GoodReads” dataset aside and then monitor whether the system is able to make correct predictions. We compared our results with the predictions made by GoodReads website.

We performed unit and integrated testing on project after completion of each component. During initial testing phase we found out that the ratings given to each book has to be optimised for getting the correct recommendation of books.

#### Content Based Filtering (Tag Based)

Results of our recommendation system:

```
In [94]: topRecommendations('The Catcher in the Rye').head(10)
Out[94]: 350          The Great Gatsby
          198          To Kill a Mockingbird
          87          Of Mice and Men
          326         The Grapes of Wrath
          1209       A Tree Grows in Brooklyn
          4224          The Chosen
          1048       Death of a Salesman
          982          Housekeeping
          174       The Old Man and the Sea
          Name: title, dtype: object
```

Fig 8.1


# BOOK RECOMMENDATION SYSTEM

Results of “GoodReads” website:

The screenshot displays the GoodReads website interface. At the top, there's a navigation bar with links for Home, My Books, Browse, Community, and a search bar. The main heading is "Books similar to The Catcher in the Rye". Below this, a list of recommended books is shown in two columns. Each book entry includes a cover image, the title, author, a star rating, the number of ratings, a brief synopsis, and a "Want to Read" button. The books listed are: "The Catcher in the Rye" by J.D. Salinger, "The Great Gatsby" by F. Scott Fitzgerald, "A Tree Grows in Brooklyn" by Betty Smith, "To Kill a Mockingbird" by Harper Lee, "Lord of the Flies" by William Golding, "Of Mice and Men" by John Steinbeck, "The Old Man and the Sea" by Ernest Hemingway, "1984" by George Orwell, and "The Grapes of Wrath" by John Steinbeck. A section titled "Goodreads members who liked this book also liked:" is also visible, showing "Animal Farm" by George Orwell.


goodreads Home My Books Browse Community Search Books

### Books similar to The Catcher in the Rye




**The Catcher in the Rye**  
by J.D. Salinger  
★★★★☆ 4.55 avg. rating · 2,002,270 Ratings  
The hero-narrator of *The Catcher in the Rye* is an adolescent child of sixteen, a native New Yorker named Holden Caulfield. Through circumstances that tend to prefigure adult, second-hand descriptions, he be...  
[Want to Read](#) [Rate](#) [1](#) [2](#) [3](#) [4](#) [5](#)


Goodreads members who liked this book also liked:




**The Great Gatsby**  
by F. Scott Fitzgerald  
★★★★☆ 4.16 avg. rating · 4,041,304 Ratings  
Advanced Cover Edition ISBN: 0743273567 ISBN13: 9780743273565  
The Great Gatsby, F. Scott Fitzgerald's third book, stands as the supreme achievement of his career. This exemplary novel of the Jazz Age...  
[Want to Read](#) [Rate](#) [1](#) [2](#) [3](#) [4](#) [5](#)




**A Tree Grows in Brooklyn**  
by Betty Smith  
★★★★☆ 4.35 avg. rating · 202,272 Ratings  
The beloved American classic about a young girl's coming-of-age at the turn of the century, Betty Smith's *A Tree Grows in Brooklyn* is a poignant and moving tale filled with compassion and gritty realism...  
[Want to Read](#) [Rate](#) [1](#) [2](#) [3](#) [4](#) [5](#)




**To Kill a Mockingbird**  
by Harper Lee  
★★★★☆ 4.07 avg. rating · 4,074,012 Ratings  
The unforgettable story of a childhood in a sleepy Southern town and the crisis of conscience that rocked it, *To Kill a Mockingbird* became both an instant bestseller and a critical success when it was...  
[Want to Read](#) [Rate](#) [1](#) [2](#) [3](#) [4](#) [5](#)




**Lord of the Flies**  
by William Golding  
★★★★☆ 3.68 avg. rating · 1,733,094 Ratings  
At the dawn of the next world war, a plane crashes on an uninhabited island, leaving a group of schoolboys. At first, with no adult supervision, their behavior is so excellent that they begin to...  
[Want to Read](#) [Rate](#) [1](#) [2](#) [3](#) [4](#) [5](#)




**Of Mice and Men**  
by John Steinbeck  
★★★★☆ 3.67 avg. rating · 1,201,074 Ratings  
The compelling story of two outsiders striving to find their place in an unforgiving world.  
Cofounders in search of work, George and his simple-minded friend Lennie have nothing in the world except each other...  
[Want to Read](#) [Rate](#) [1](#) [2](#) [3](#) [4](#) [5](#)



**The Old Man and the Sea**  
by Ernest Hemingway  
★★★★☆ 3.77 avg. rating · 725,120 Ratings  
[Designer's note: An alternate cover edition can be found here]  
This short novel, already a modern classic, is the superbly told, highly stylized story of a Cuban fisherman in the Gulf Stream and the giant Marlin...  
[Want to Read](#) [Rate](#) [1](#) [2](#) [3](#) [4](#) [5](#)




**1984**  
by George Orwell  
★★★★☆ 4.58 avg. rating · 1,021,422 Ratings  
Among the seminal books of the 20th century, *Nineteen Eighty-Four* is a late work that grows more haunting as its futuristic prophecy becomes more real. Published in 1949, the book offers political...  
[Want to Read](#) [Rate](#) [1](#) [2](#) [3](#) [4](#) [5](#)



**The Grapes of Wrath**  
by John Steinbeck  
★★★★☆ 4.00 avg. rating · 275,123 Ratings  
The Pulitzer Prize-winning epic of the Great Depression, a book that galvanized—and sometimes outraged—millions of readers.  
First published in 1939, Steinbeck's Pulitzer Prize-winning epic of the Great...  
[Want to Read](#) [Rate](#) [1](#) [2](#) [3](#) [4](#) [5](#)

Goodreads members who liked this book also liked:



**Animal Farm**  
by George Orwell  
★★★★☆ 4.05 avg. rating · 342,344 Ratings  
[Designer's note: There is an Alternate Cover Edition for this edition of this book here]  
A farm is taken over by its overworked, mistreated animals. With flaming idealism and stirring slogans, they set...  
[Want to Read](#) [Rate](#) [1](#) [2](#) [3](#) [4](#) [5](#)

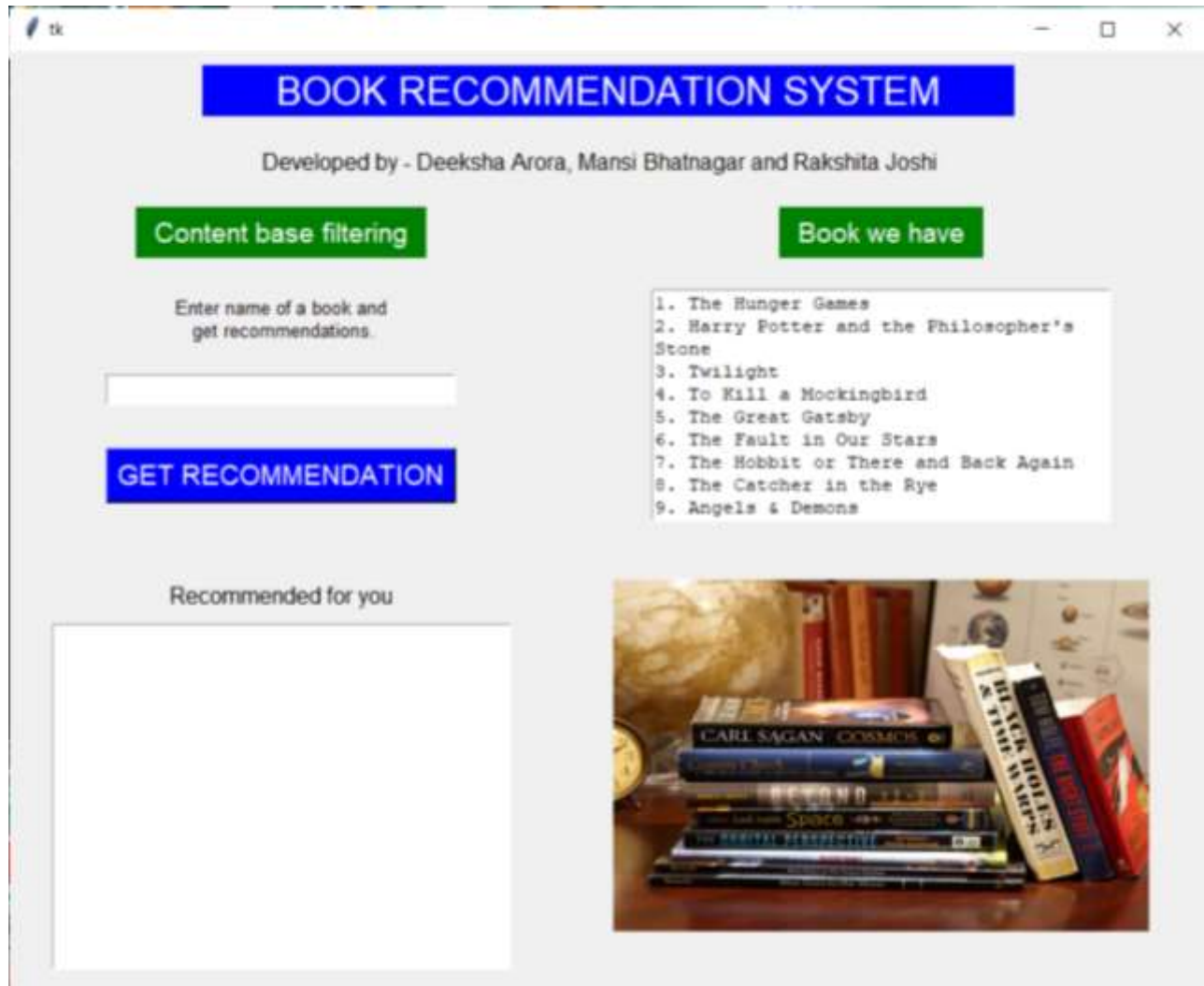
Fig 8.2

# BOOK RECOMMENDATION SYSTEM

---

## 8.2 Results of our Recommendation System:

1. User visits the Desktop app where he/she can select from a list of books.



## BOOK RECOMMENDATION SYSTEM

---

2. User enters the book of his choice Clicks on “Get Recommendation” button to get recommendations for books.

tk

— □ ×

# BOOK RECOMMENDATION SYSTEM

Developed by - Deeksha Arora, Mansi Bhatnagar and Rakshita Joshi

### Content base filtering

Enter name of a book and get recommendations.


  

## GET RECOMMENDATION

### Book we have

1. The Hunger Games
2. Harry Potter and the Philosopher's Stone
3. Twilight
4. To Kill a Mockingbird
5. The Great Gatsby
6. The Fault in Our Stars
7. The Hobbit or There and Back Again
8. The Catcher in the Rye
9. Angels & Demons

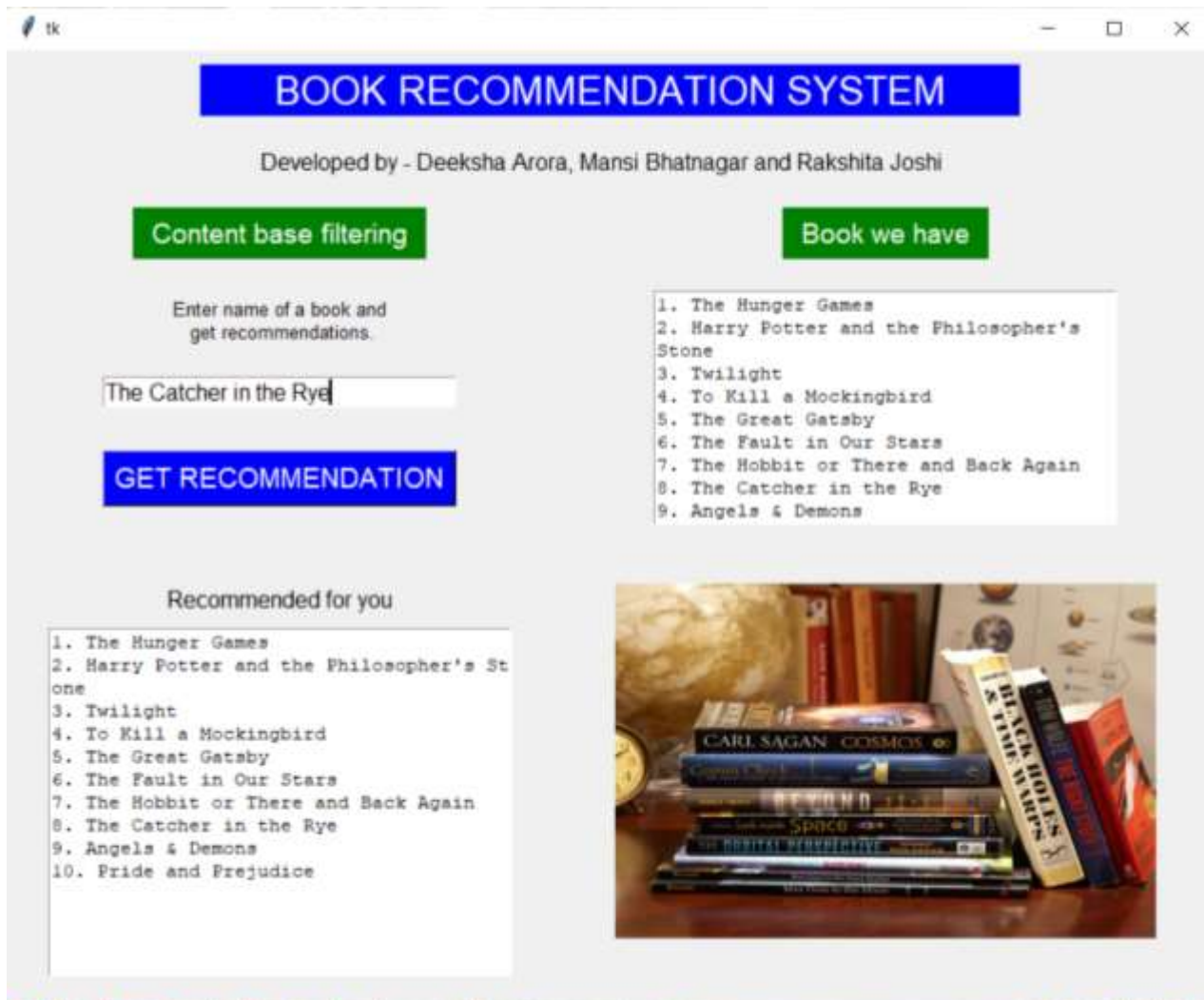
Recommended for you



## BOOK RECOMMENDATION SYSTEM

---

3. The recommended books are displayed to the users.



**CHAPTER 9**  
**CONCLUSION**



### CONCLUSION

#### **9.1 Advantages of the System**

- The System would benefit those users who have to use search engines to locate relevant content. They have to scroll through pages of results to find relevant content.
- Rather than searching for quality books, the users of this system would be directly recommended quality books matching their personal interests and preferences.
- The system would recommend quality books as it is not just dependent on the rating given by other users which could be deceiving at times.

#### **9.2 Future Enhancement**

In this project a recommendation system has been implemented based on collaborative filtering and content filtering. The system can be highly improved by making use of caching mechanisms, user clustering which will definitely boost the speed of the system, using yahoo term extraction web service to parse and get important keywords from the feedback provided by the user for an item and utilizing these keywords in context based engine. Further enhancements include storing users past history of results, contexts for future predictions.

#### **9.3 Conclusion**

The goal of most of the recommendation systems is to predict the reader's interest and recommend the books accordingly. This book recommendation system has considered many parameters like content of the book, author of the book and quality of the book by doing content filtering of ratings by the other readers. Our experience while working on this project has enhanced our skills and has contributed towards our professional and personal development.

### REFERENCES

1. [https://www.researchgate.net/publication/325792825\\_An\\_Improved\\_Online\\_Book\\_Recommender\\_System\\_using\\_Collaborative\\_Filtering\\_Algorithm](https://www.researchgate.net/publication/325792825_An_Improved_Online_Book_Recommender_System_using_Collaborative_Filtering_Algorithm)
2. <http://infolab.stanford.edu/~ullman/mmds/ch9.pdf>
3. <http://www.grouplens.org>
4. [http://en.wikipedia.org/wiki/Machine\\_learning](http://en.wikipedia.org/wiki/Machine_learning)
5. <http://taste.sourceforge.net/>
6. Stuart E. Middleton, Nigel R. Shadbolt (2004). Ontological user profiling in recommender systems. ACM Transactions on Information Systems, 54-88.
7. <http://thesaurus.reference.com/>
8. Toby Segaran (2007). Programming Collective Intelligence: Building Smart Web 2.0 Applications, Page11