MAJOR PROJECT
ON
Movie Recommendation System

Submitted in partial fulfillment of the requirement for
the award of the degree of

BCA

+

MCA

Lokesh Yadav
Enroll. No. A50549515005

Under the guidance of

Miss Poonam Sharma

Assistant Professor,

CSE

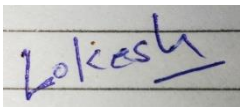
Department of Computer Science and Engineering
Amity Institute of Information Technology
Amity University Haryana
Gurgaon, India
May 2020

Amity Institute of Information Technology

Amity University Haryana

DECLARATION

I, **Lokesh Yadav**, student of Master of Computer Applications hereby declare that the project entitled **"Movie Recommendation System"** which is submitted by us to department of Amity Institute of Information and Technology, Amity University Haryana, in partial fulfillment of the requirement for the award of the degree of Master of Computer Applications, has not been previously formed the basis for the award of any degree, diploma or other similar title or recognition.



Lokesh Yadav

Enroll No.-A50549515005

Amity Institute of Information Technology

Amity University Haryana

CERTIFICATE

This is to certify that the work in the project report entitled ***"Movie Recommendation System"*** by ***Lokesh Yadav*** bearing ***Enroll. No. A50549515005*** is a bona fide record of project work carried out by her under my supervision and guidance in partial fulfillment of the requirements for the award of the degree of BCA + MCA in the Amity Institute of Information Technology, Amity University Haryana. The work was satisfactory. He has shown complete dedication to the given project work.

Date:

Miss Poonam Sharma

Associate Professor (AIIT)

Head
Department of Computer Science and Engineering
Amity Institute of information Technology
Amity University Haryana, Gurgaon

Infotech

Date: 25/01/2020

Ref: AAM/CL2020/8684

Lokesh

Amity University

Gurgaon

Dear **Lokesh,**

This is with reference to your application that we had in respect of **'Industrial Training'** sought by you with the Company.

In this connection, we are pleased to inform you that it has been decided to take you as **"Python+Data Science"** for a period of approx **6 Months** i.e. **Jan 2020 to June 2020**.

For this training period, you will be working under the guidance of **Mr. Tushar** or any other person deputed by him to impart training to you and you will work in accordance with the directions given to you from time to time by the person under whom you may be directed to work.

During the training period, you will be governed by the rules of the company as are applicable to Trainees.

Best Regards,

For AAM INFOTECH PVT. LTD.

Authorised Signatory

Auth. Signatory

Address

Second Floor, 180872, Old DLF Sector 14, Gurgaon (H.R.)
Tel.: 0124-4219095
Email: info@aaminfotech.com
Web: www.aaminfotech.com

ACKNOWLEDGMENT

First and the foremost I would like to thank to my almighty for giving me courage to thanks to my teacher, friends and other sources that gave an unending support and helped me in numerous ways from the first stage of my term assignment conceived. I would also like to thank my family members for their whole hearted support and cooperation. I duly acknowledge the contribution of Ms. Poonam Sharma for invaluable help. Coding Movie Recommendation System is an uphill task and would have not been possible without proper and timely assistance of her. I would also thanks to all my friends for forwarding their suggestions.

ABSTRACT

A recommendation engine filters the data using different algorithms and recommends the most relevant items to users. It first catches the past conduct of a client and dependent on that, suggests items which the clients may probably purchase. On the off chance that a totally new client visits an online business website, that webpage won't have any previous history of that client. So how does the site approach prescribing items to the client in such a situation? One potential arrangement could be to suggest the top of the line items, for example the items which are high sought after. Another conceivable arrangement could be to suggest the items which would carry the most extreme benefit to the business. Three primary methodologies are utilized for our recommender frameworks. One is Demographic Filtering i.e. they offer summed up proposals to each client, in view of film fame as well as kind. The System prescribes similar motion pictures to clients with comparable segment highlights. Since every client is diverse, this methodology is viewed as excessively straightforward. The fundamental thought behind this framework is that films that are increasingly famous and widely praised will have a higher likelihood of being preferred by the normal crowd. Second is content-based sifting, where we attempt to profile the client's intrigues utilizing data gathered, and suggest things dependent on that profile. The other is cooperative sifting, where we attempt to aggregate comparative clients and use data about the gathering to make suggestions to the client.

CONTENTS

LIST OF FIGURES

Chapter 1
Introduction

A recommendation system is a type of information filtering system which attempts to predict the preferences of a user, and make suggests based on these preferences. There is a wide assortment of uses for suggestion frameworks. These have gotten progressively well known in the course of the most recent couple of years and are currently used in most online stages that we use. The substance of such stages shifts from motion pictures, music, books and recordings, to companions and stories via web-based networking media stages, to items on internet business sites, to individuals on expert and dating sites, to query items returned on Google. Regularly, these frameworks can gather data about a client's decisions, and can utilize this data to improve their proposals later on. For instance, Facebook can screen your connection with different stories on your channel so as to realize what sorts of stories claim to you. Some of the time, the recommender frameworks can make enhancements dependent on the exercises of an enormous number of individuals. For instance, if Amazon sees that countless clients who purchase the most recent Apple MacBook additionally purchase a USB-C-to USB Adapter, they can prescribe the Adapter to another client who has quite recently added a MacBook to his truck. Because of the advances in recommender frameworks, clients continually anticipate great suggestions. They have a low edge for administrations that can't make proper recommendations. In the event that a music gushing application can't foresee and play music that the client likes, at that point the client will just quit utilizing it. This has prompted a high accentuation by tech organizations on improving their proposal frameworks. Be that as it may, the issue is surprisingly mind boggling. Each client has various inclinations and preferences. Moreover, even the flavor of a solitary client can fluctuate contingent upon countless variables, for example, disposition, season, or kind of movement the client is doing. For instance, the kind of music one might want to hear while practicing varies incredibly from the sort of music he'd tune in to when preparing supper. Another issue that proposal frameworks need to understand is the investigation versus misuse issue. They should investigate new areas to find increasingly about the client, while as yet benefiting as much as possible from what is as of now thought about of the client. Three principle approaches are utilized for our recommender frameworks. One is Demographic Filtering i.e. they offer summed up suggestions to each client, in light of

film notoriety or potentially type. The System prescribes similar motion pictures to clients with comparable segment highlights. Since every client is distinctive, this methodology is viewed as excessively basic. The fundamental thought behind this framework is that films that are increasingly mainstream and widely praised will have a higher likelihood of being enjoyed by the normal crowd. Second is content-based separating, where we attempt to profile the client's intrigues utilizing data gathered, and suggest things dependent on that profile. The other is collective separating, where we attempt to gather comparative clients and use data about the gathering to make suggestions to the client.

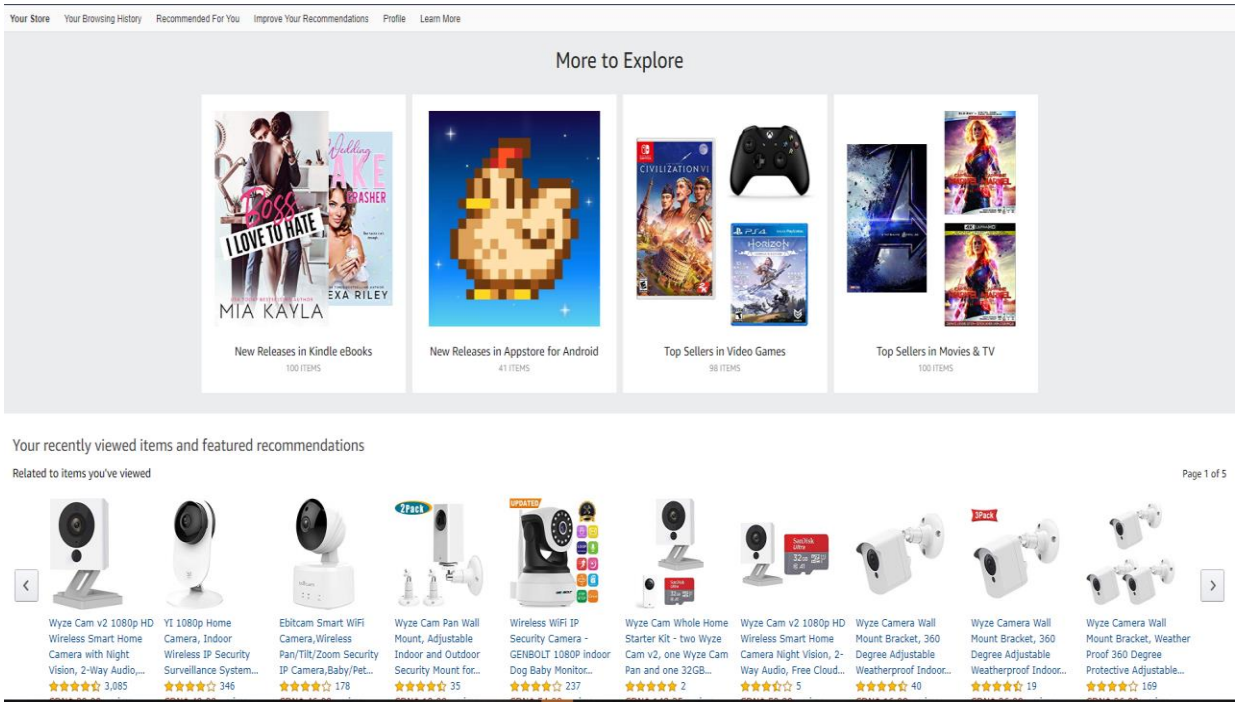


Figure 1: Recommended Products

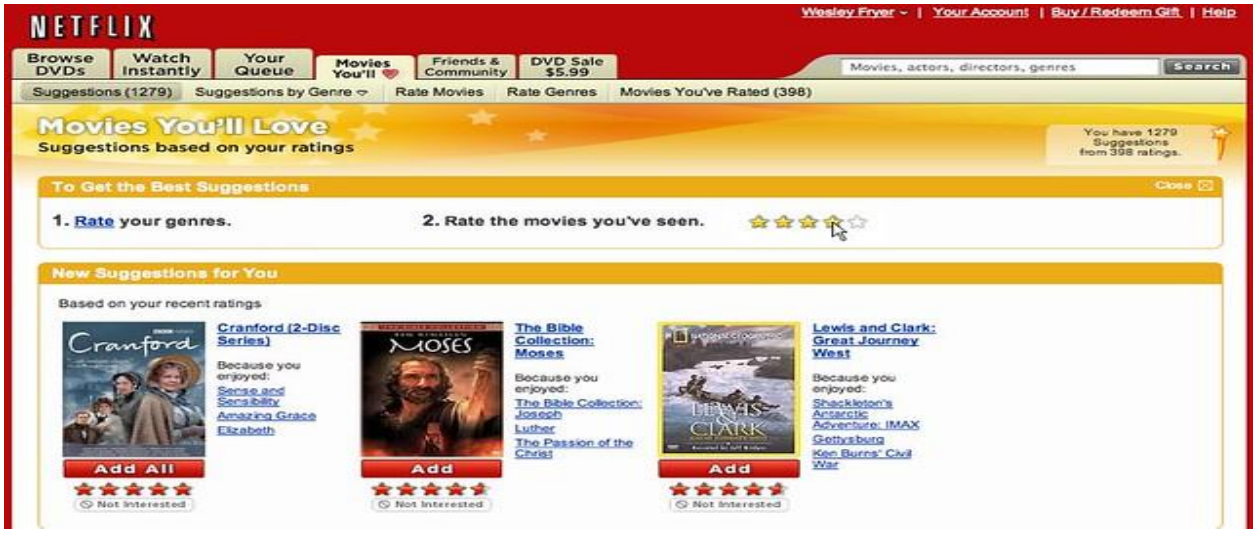


Figure 2: Netflix

1.1 Objective

Movie recommendation systems provide a mechanism to assist users in classifying users with similar interests. This System focuses on the movie recommendation systems whose primary objective is to suggest a recommender system through data clustering and computational intelligence. The purpose of a recommendation system basically is to search for content that would be interesting to an individual. Moreover, it involves a number of factors to create personalized lists of useful and interesting content specific to each user/individual. Recommendation systems are Artificial Intelligence based algorithms that skim through all possible options and create a customized list of items that are interesting and relevant to an individual. These results are based on their profile; search/browsing history, what other people with similar traits/demographics are watching, and how likely are you to watch those movies. This is achieved through predictive modeling and heuristics with the data available.

1.2 Fundamentals of Machine Learning

Machine learning (**ML**) is the study of computer algorithms that improve automatically through experience [1]. It is seen as a subset of artificial intelligence. Machine learning algorithms build a mathematical model based on sample data, known as "training data", in order to make predictions or decisions without being explicitly programmed to do so [2][3]. Machine learning algorithms are used in a wide variety of applications, such as email

filtering and computer vision, where it is difficult or infeasible to develop conventional algorithms to perform the needed tasks.

Machine learning is closely related to computational statistics, which focuses on making predictions using computers. The study of mathematical optimization delivers methods, theory and application domains to the field of machine learning. Data mining is a related field of study, focusing on exploratory data analysis through unsupervised learning [4]. In its application across business problems, machine learning is also referred to as predictive analytics.

Machine learning involves computers discovering how they can perform tasks without being explicitly programmed to do so. For simple tasks assigned to computers, it is possible to program algorithms telling the machine how to execute all steps required to solve the problem at hand; on the computer's part, no learning is needed. For more advanced tasks, it can be challenging for a human to manually create the needed algorithms. In practice, it can turn out to be more effective to help the machine develop its own algorithm, rather than have human programmers specify every needed step [3].

The discipline of machine learning employs various approaches to help computers learn to accomplish tasks where no fully satisfactory algorithm is available. In cases where vast numbers of potential answers exist, one approach is to label some of the correct answers as valid. This can then be used as training data for the computer to improve the algorithm(s) it uses to determine correct answers. For example, to train a system for the task of digital character recognition, the MNIST dataset has often been used.

Early classifications for machine learning approaches sometimes divided them into three broad categories, depending on the nature of the "signal" or "feedback" available to the learning system.

Supervised learning: The computer is presented with example inputs and their desired outputs, given by a "teacher", and the goal is to learn a general rule that maps inputs to outputs. Unsupervised learning: No labels are given to the learning algorithm, leaving it on its own to find structure in its input. Unsupervised learning can be a goal in itself (discovering hidden patterns in data) or a means towards an end (feature learning). Reinforcement learning: A computer program interacts with a dynamic environment in which it

must perform a certain goal [1] (such as driving a vehicle or playing a game against an opponent) As it navigates its problem space, the program is provided feedback that's analogous to rewards, which it tries to maximize.

Other approaches or processes have since developed that don't fit neatly into this three-fold categorization, and sometimes more than one is used by the same machine learning system. For example topic modeling, dimensionality reduction or meta learning. As of 2020, deep learning had become the dominant approach for much ongoing work in the field of machine learning.

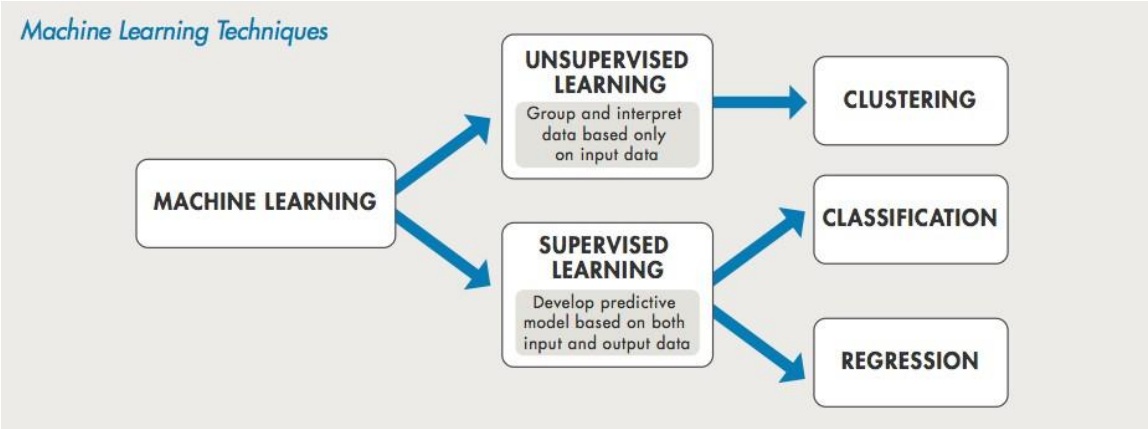


Figure 3: Machine Learning Techniques

Chapter 2

Literature Review

MOVREC is a movie recommendation system presented by D.K. Yadav et al. based on collaborative filtering approach. Collaborative filtering makes use of information provided by user. That information is analyzed and a movie is recommended to the users which are arranged with the movie with highest rating first [33].

Luis M Capos et al has analyzed two traditional recommender systems i.e. content based filtering and collaborative filtering. As both of them have their own drawbacks he proposed a new system which is a combination of Bayesian network [34] and collaborative filtering.

A hybrid system has been presented by Harpreet Kaur et al. The system uses a mix of content as well as collaborative filtering algorithm. The context of the movies is also considered while recommending. The user - user relationship as well as user - item relationship plays a role in the recommendation.

The user specific information or item specific information is clubbed to form a cluster by Utkarsh Gupta et al. using chameleon. This is an efficient technique based on Hierarchical clustering for recommender system [35]. To predict the rating of an item voting system is used. The proposed system has lower error and has better clustering of similar items.

Urszula Kużelewska et al. proposed clustering as a way to deal with recommender systems. Two methods of computing cluster representatives were presented and evaluated. Centroid-based solution and memory-based collaborative filtering methods were used as a basis for comparing effectiveness of the proposed two methods. The result was a significant increase in the accuracy of the generated recommendations when compared to just centroid-based method [36]. Costin-Gabriel Chiru et al. proposed Movie Recommender, a system which uses the information known about the user to provide movie recommendations. This system attempts to solve the problem of unique recommendations which results from ignoring the data specific to the user. The psychological profile of the user, their watching history and the data involving movie scores from other websites is collected. They are based on aggregate similarity calculation. The system is a hybrid model which uses both content based filtering and collaborative filtering.

To predict the difficulty level of each case for each trainee Hongli LIn et al. proposed a method called contentboosted collaborative filtering (CBCF).The algorithm is divided into two stages, First [37]being the content-based filtering that improves the existing trainee case ratings data and the second being collaborative filtering that provides the final predictions. The CBCF algorithm involves the advantages of both CBF and CF [38], while at the same time, overcoming both their disadvantages.

Chapter 3

Proposed Model

With our aim being to predict which movie is similar to the listed movie, we have outlined a simple model to come with the most accurate predictions. The first objective was to attain a dataset of numerical values of various instances. Feature selection in the form of Principal Component Analysis is used to reduce dimensionality of the dataset. The models are trained again by means of training and testing after PCA is applied and finally compared the results with that of the previous results we reached without PCA.

The workflow below outlines a basic review of the entire thesis:

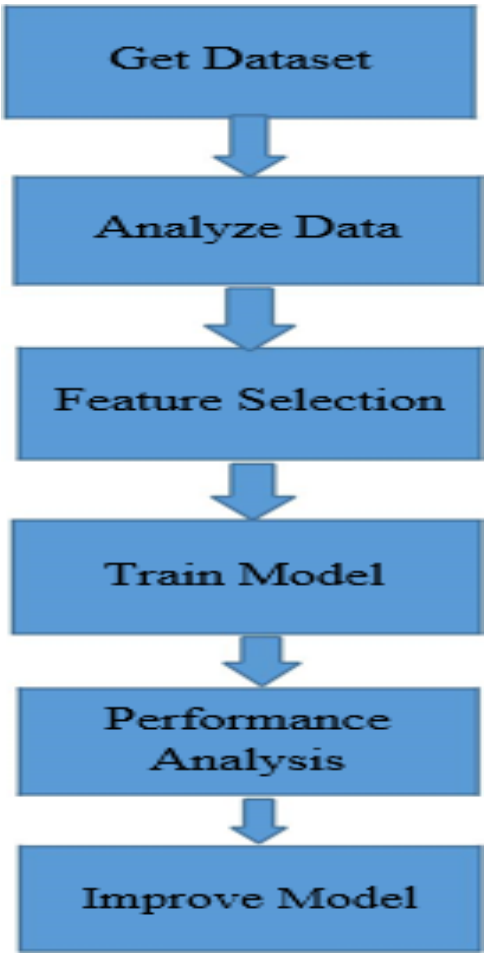


Figure 4: Workflow

Data requirements

The data are necessary as inputs to the analysis, which is specified based upon the requirements of those directing the analysis or customers (who will use the finished product of the analysis). The general type of entity upon which the data will be collected is referred to as an experimental unit (e.g., a person or population of people). Specific variables regarding a population (e.g., age and income) may be specified and obtained. Data may be numerical or categorical (i.e., a text label for numbers) [5].

Data collection

Data are collected from a variety of sources. The requirements may be communicated by analysts to custodians of the data, such as information technology personnel within an organization. The data may also be collected from sensors in the environment, such as traffic cameras, satellites, recording devices, etc. It may also be obtained through interviews, downloads from online sources, or reading documentation [5].

Data Processing

Data initially obtained must be processed or organized for analysis. For instance, these may involve placing data into rows and columns in a table format (i.e., structured data) for further analysis, such as within a spreadsheet or statistical software [5].

Data cleaning

Once processed and organized, the data may be incomplete, contain duplicates, or contain errors. The need for data cleaning will arise from problems in the way that data are entered and stored. Data cleaning is the process of preventing and correcting these errors. Common tasks include record matching, identifying inaccuracy of data, and overall quality of existing data, reduplication, and column segmentation [7]. Such data problems can also be identified through a variety of analytical techniques. For example, with financial information, the totals for particular variables may be compared against separately published numbers believed to be reliable [8]. Unusual amounts above or below pre-determined thresholds may also be reviewed. There are several types of data cleaning that depend on the type of data such as phone numbers, email addresses, employers etc. Quantitative data methods for outlier detection can be used to

get rid of likely incorrectly entered data. Textual data spell checkers can be used to lessen the amount of mistyped words, but it is harder to tell if the words themselves are correct [9].

Exploratory data analysis

Once the data are cleaned, it can be analyzed. Analysts may apply a variety of techniques referred to as exploratory data analysis to begin understanding the messages contained in the data[11][10]. The process of exploration may result in additional data cleaning or additional requests for data, so these activities may be iterative in nature. Descriptive statistics, such as the average or median, may be generated to help understand the data. Data visualization may also be used to examine the data in graphical format, to obtain additional insight regarding the messages within the data [5].

Modeling and algorithms

Mathematical formulas or models called algorithms may be applied to the data to identify relationships among the variables, such as correlation or causation. In general terms, models may be developed to evaluate a particular variable in the data based on other variable(s) in the data, with some residual error depending on model accuracy (i.e., Data = Model + Error) [6].

Inferential statistics includes techniques to measure relationships between particular variables. For example, regression analysis may be used to model whether a change in advertising (independent variable X) explains the variation in sales (dependent variable Y). In mathematical terms, Y (sales) is a function of X (advertising). It may be described as $Y = aX + b +$ error, where the model is designed such that a and b minimize the error when the model predicts Y for a given range of values of X. Analysts may attempt to build models that are descriptive of the data to simplify analysis and communicate results [6].

Data product

A data product is a computer application that takes data inputs and generates outputs, feeding them back into the environment. It may be based on a model or algorithm. An example is an application that analyzes data about customer purchasing history and recommends other purchases the customer might enjoy [8].

Chapter 4

4.1 Dataset

MovieLens is a web-based recommender system and virtual community that recommends movies for its users to watch, based on their film preferences using collaborative filtering of members' movie ratings and movie reviews. It contains about 11 million ratings for about 8500 movies. Movie Lens was created in 1997 by Group Lens Research, a research lab in the Department of Computer Science and Engineering at the University of Minnesota, in order to gather research data on personalized recommendations.
Movie Lens data sets were collected by the Group Lens Research Project
at the University of Minnesota.

This data set consists of:

* 100,000 ratings (1-5) from 943 users on 1682 movies.
* Each user has rated at least 20 movies.
* Simple demographic info for the users (age, gender, occupation, zip)

The data was collected through the Movie Lens web site (movielens.umn.edu).
This data has been cleaned up – users who had less than 20 ratings or did not have complete demographic information were removed from this data set. Detailed descriptions of the data file can be found at the end of this file [12].

4.2 Technology Used:

4.2.1 Python:

Python is an interpreted, high-level, general-purpose programming language. Created by Guido van Rossum and first released in 1991, Python's design philosophy emphasizes code readability with its notable use of significant whitespace. Its language constructs and object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects [13].

Python is dynamically typed and garbage-collected. It supports multiple programming paradigms, including structured (particularly, procedural), object-oriented, and functional programming. Python is often described as a "batteries included" language due to its comprehensive standard library [14].

Python was conceived in the late 1980s as a successor to the ABC language. Python 2.0, released in 2000, introduced features like list comprehensions and a garbage collection system capable of collecting reference cycles. Python 3.0, released in 2008, was a major revision of the language that is not completely backward-compatible, and much Python 2 code does not run unmodified on Python 3.

The Python 2 language was officially discontinued in 2020 (first planned for 2015), and "Python 2.7.18 is the last Python 2.7 release and therefore the last Python 2 release [15]." No more security patches or other improvements will be released for it[16][17]. With Python 2's end-of-life, only Python 3.5.x and later are supported.

Python interpreters are available for many operating systems. A global community of programmers develops and maintains CPython, an open source reference[18] implementation. A non-profit organization, the Python Software Foundation, manages and directs resources for Python and CPython development.

Python was conceived in the late 1980s[19] by Guido van Rossum at Centrum Wiskunde & Informatica (CWI) in the Netherlands as a successor to the ABC language (itself inspired by SETL)[20] , capable of exception handling and interfacing with the Amoeba operating system [13]. Its implementation began in December 1989 [21]. Van Rossum shouldered sole responsibility for the project, as the lead developer, until 12 July 2018, when he announced his "permanent vacation" from his responsibilities as Python's *Benevolent Dictator For Life*, a title the Python community bestowed upon him to reflect his long-term commitment as the project's chief decision-maker[22]. He now shares his leadership as a member of a five-person steering council [23][24]. In January 2019, active Python core developers elected Brett Cannon, Nick Coghlan, Barry Warsaw, Carol Willing and Van Rossum to a five-member "Steering Council" to lead the project [25].

Python uses dynamic typing and a combination of reference counting and a cycle-detecting garbage collector for memory management. It also features dynamic name resolution (late binding), which binds method and variable names during program execution.

Python is meant to be an easily readable language. Its formatting is visually uncluttered, and it often uses English keywords where other languages use punctuation. Unlike many other languages,

it does not use curly brackets to delimit blocks, and semicolons after statements are optional. It has fewer syntactic exceptions and special cases than C or Pascal [26].

Python uses whitespace indentation, rather than curly brackets or keywords, to delimit blocks. An increase in indentation comes after certain statements; a decrease in indentation signifies the end of the current block [27]. Thus, the program's visual structure accurately represents the program's semantic structure. This feature is sometimes termed the off-side rule, which some other languages share, but in most languages indentation doesn't have any semantic meaning.

Python interpreters are available for many operating systems. A global community of programmers develops and maintains CPython, an open source reference implementation. A non-profit organization, the Python Software Foundation, manages and directs resources for Python and CPython development.

4.2.2 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 is a nonprofit organization created to "develop open-source software, open-standards, and services for interactive computing across dozens of programming languages". Spun-off from IPython in 2014 by Fernando Pérez, Project Jupyter supports execution environments in several dozen languages. Project Jupyter's name is a reference to the three core programming languages supported by Jupyter, which are Julia, Python and R, and also a homage to Galileo's notebooks recording the discovery of the moons of Jupiter. Project Jupyter has developed and supported the interactive computing products Jupyter Notebook, JupyterHub, and JupyterLab, the next-generation version of Jupyter Notebook [28].

In 2014, Fernando Pérez announced a spin-off project from IPython called Project Jupyter. IPython continued to exist as a Python shell and a kernel for Jupyter, while the notebook and other language-agnostic parts of IPython moved under the Jupyter name. Jupyter is language agnostic and it supports execution environments (aka kernels) in several dozen languages among which are Julia, R, Haskell, Ruby, and of course Python (via the IPython kernel).

In 2015, GitHub and the Jupyter Project announced native rendering of Jupyter notebooks file format (.ipynb files) on the GitHub platform [29][30].

Jupyter Notebook (formerly IPython Notebooks) is a web-based interactive computational environment for creating Jupyter notebook documents. The "notebook" term can colloquially make reference to many different entities, mainly the Jupyter web application, Jupyter Python web server, or Jupyter document format depending on context. A Jupyter Notebook document is a JSON document, following a versioned schema, and containing an ordered list of input/output cells which can contain code, text (using Markdown), mathematics, plots and rich media, usually ending with the ".ipynb" extension.

A Jupyter Notebook can be converted to a number of open standard output formats (HTML, presentation slides, LaTeX, PDF, ReStructuredText, Markdown, Python) through "Download As" in the web interface, via the nbconvert library or "jupyter nbconvert" command line interface in a shell. To simplify visualisation of Jupyter notebook documents on the web, the nbconvert library is provided as a service through NbViewer which can take a URL to any publicly available notebook document, convert it to HTML on the fly and display it to the user.

Jupyter Notebook can connect to many kernels to allow programming in many languages. By default Jupyter Notebook ships with the IPython kernel. As of the 2.3 release (October 2014), there are currently 49 Jupyter-compatible kernels for many programming languages, including Python, R, Julia and Haskell.

The Notebook interface was added to IPython in the 0.12 release (December 2011), renamed to Jupyter notebook in 2015 (IPython 4.0 – Jupyter 1.0). Jupyter Notebook is similar to the notebook interface of other programs such as Maple, Mathematica, and SageMath, a computational interface style that originated with Mathematica in the 1980s. According to *The Atlantic*, Jupyter interest overtook the popularity of the Mathematica notebook interface in early 2018.

A Jupyter kernel is a program responsible for handling various types of requests (code execution, code completions, inspection), and providing a reply. Kernels talk to the other components of Jupyter using ZeroMQ over the network, and thus can be on the same or remote machines. Unlike many other Notebook-like interfaces, in Jupyter, kernels are not aware that they are attached to a specific document, and can be connected to many clients at once. Usually kernels allow execution of only a single language, but there are a couple of exceptions.

Chapter 5

Approaches

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

5.1 Content-based Filtering Systems:

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

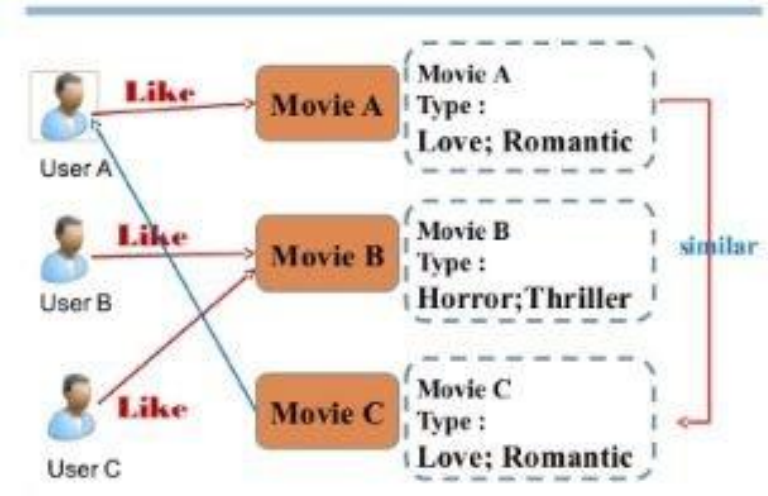


Figure 5: Content Based Filtering

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

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

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

- Get the index of the movie given its title.
- Get the list of cosine similarity scores for that particular movie with all movies.
- Convert it into a list of tuples where the first element is its position and the second is the similarity score.
- Sort the aforementioned list of tuples based on the similarity scores; that is, the second element.
- Get the top 10 elements of this list. Ignore the first element as it refers to self (the movie most similar to a particular movie is the movie itself).
- Return the titles corresponding to the indices of the top elements.

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

Advantages of content-based filtering are:

- They capable of recommending unrated items.

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

Disadvantages of content-based filtering are:

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

5.2 Collaborative filtering based systems:

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

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

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

5.2.1 User based filtering

These systems recommend products to a user that similar users have liked. For measuring the similarity between two users we can either use Pearson correlation or cosine similarity. This filtering technique can be illustrated with an example. In the following matrix's, each row represents a user, while the columns correspond to different movies except the last one which records the similarity between that user and the target user. Each cell represents the rating that the

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

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

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

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

Figure 6: User Based Filtering Matrix

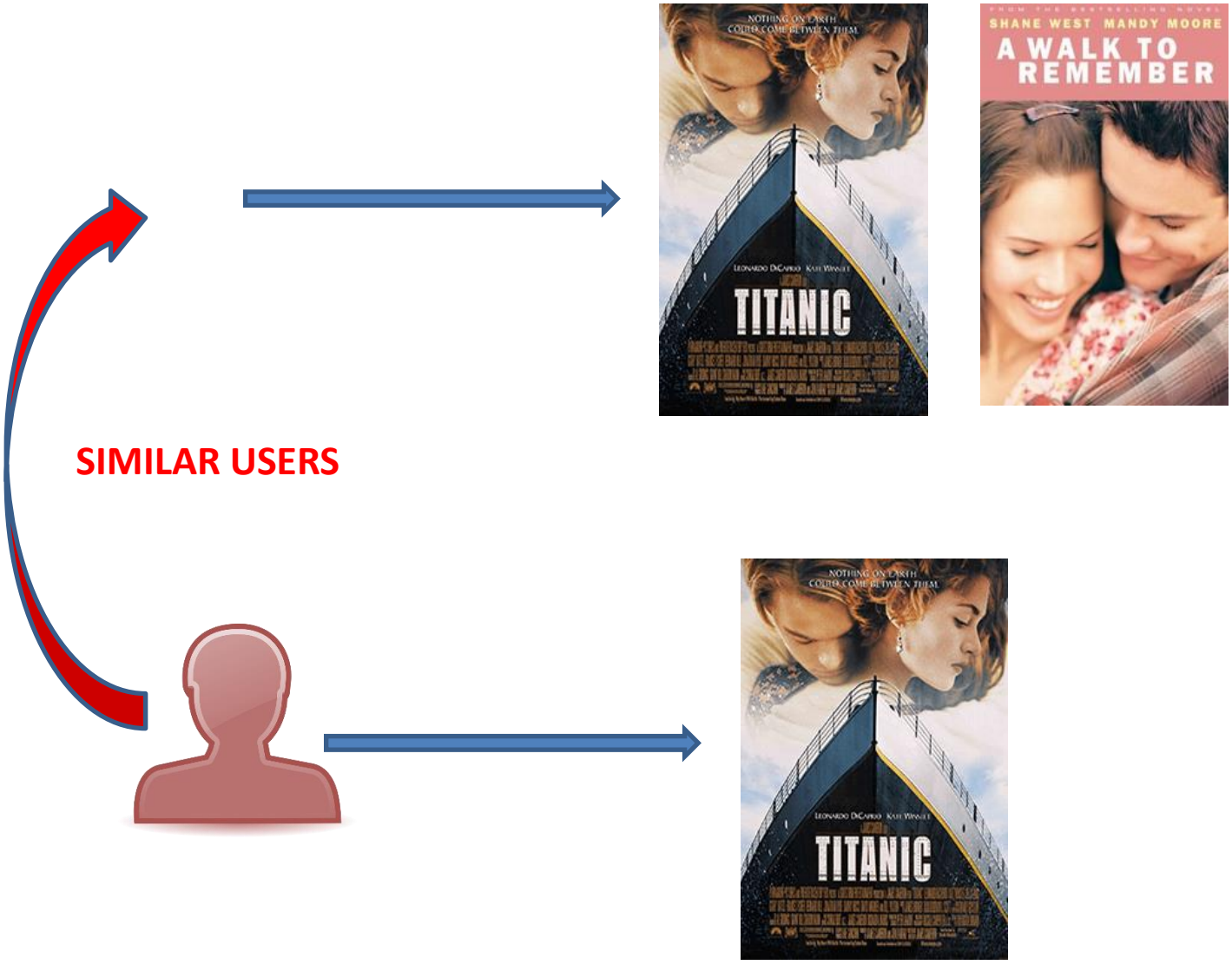


Figure 7: User Based Filtering

5.2.2 Item Based Collaborative Filtering

Instead of measuring the similarity between users, the item-based CF recommends items based on their similarity with the items that the target user rated. Likewise, the similarity can be computed with Pearson Correlation or Cosine Similarity. The major difference is that, with item-based collaborative filtering, we fill in the blank vertically, as oppose to the horizontal manner that user-based CF does. The following table shows how to do so f or the movie.

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

Advantages of collaborative filtering based systems:

- Itis dependent on the relation between users which implies that it is content-independent.
- CF recommender systems can suggest serendipitous items by observing similar-minded people's behavior.
- They can make real quality assessment of items by considering other peoples experience

Disadvantages of collaborative filtering are:

- Early rater problem: Collaborative filtering systems cannot provide recommendations for new items since there are no user ratings on which to base a prediction.
- Gray sheep: In order for CF based system to work, group with similar characteristics are needed. Even if such groups exist, it will be very difficult to recommend users who do not consistently agree or disagree to these groups.
- Sparsity problem: In most cases, the amount of items exceed the number of users by a great

margin which makes it difficult to find items that are rated by enough people.

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

Figure 8: Item Based Filtering Matrix

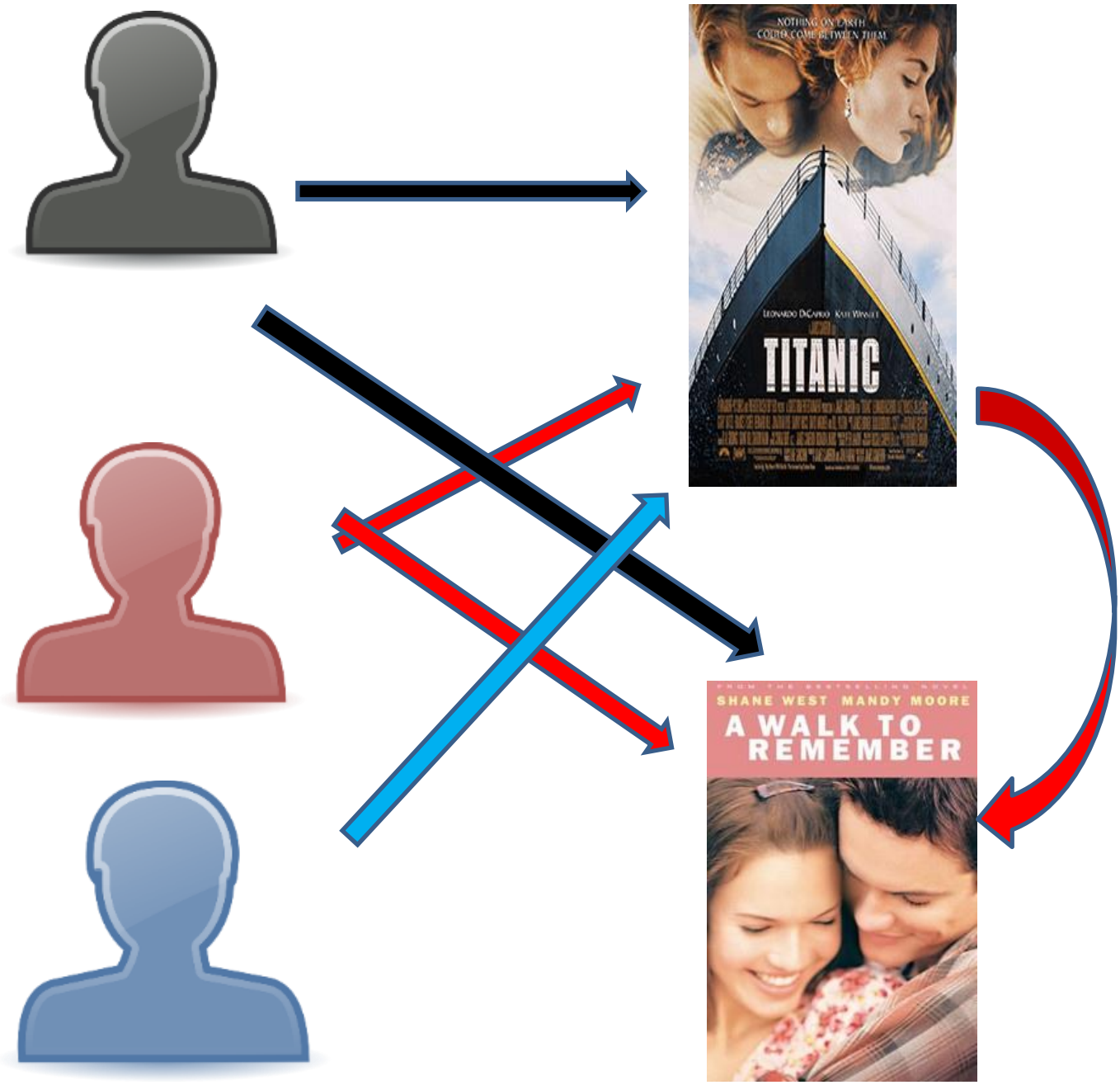


Figure 9: Item Based Filtering

Chapter 6

SCREENSHOTS

PROBLEM STATEMENT

- This notebook implements a movie recommender system.
- Recommender systems are used to suggest movies or songs to users based on their interest or usage history.
- For example, Netflix recommends movies to watch based on the previous movies you've watched.
- In this example, we will use Item-based Collaborative Filter

- Dataset MovieLens: https://grouplens.org/datasets/movielens/100k/
- Photo Credit: https://pxhere.com/en/photo/1588369

Figure 10: Problem Statement

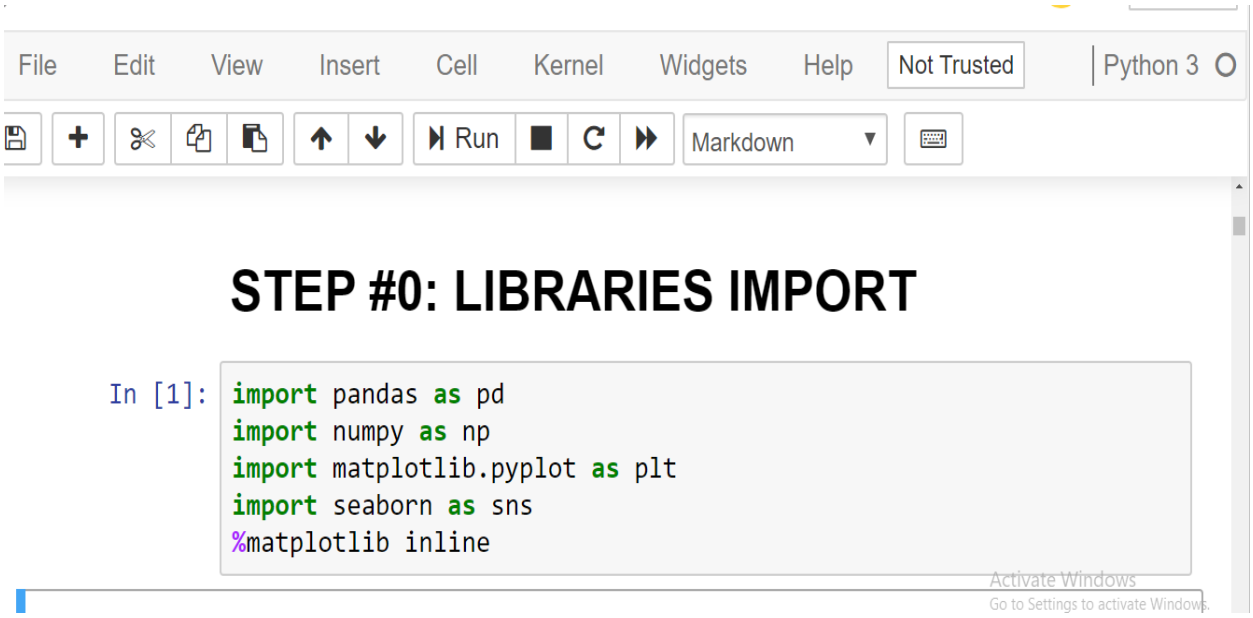


Figure 11: Required Library

STEP #1: IMPORT DATASET

```
[2]: # Two datasets are available, let's load the first one:
movie_titles_df = pd.read_csv("Movie_Id_Titles")
movie_titles_df.head(20)
```

[2]:

	item_id	title
0	1	Toy Story (1995)
1	2	GoldenEye (1995)
2	3	Four Rooms (1995)
3	4	Get Shorty (1995)
4	5	Copycat (1995)
5	6	Shanghai Triad (Yao a yao yao dao waipo qiao) ...
6	7	Twelve Monkeys (1995)
7	8	Babe (1995)
8	9	Dead Man Walking (1995)
9	10	Richard III (1995)
10	11	Seven (Se7en) (1995)
11	12	Usual Suspects, The (1995)
12	13	Mighty Aphrodite (1995)
13	14	Postino, Il (1994)
14	15	Mr. Holland's Opus (1995)
15	16	French Twist (Gazon maudit) (1995)
16	17	From Dusk Till Dawn (1996)
17	18	White Balloon, The (1995)

Figure 12: Import Dataset

```
# Let's load the second one!
movies_rating_df = pd.read_csv('u.data', sep='\t', names=['user_id', 'item_id', 'rating', 'timestamp'])
```

```
movies_rating_df.head(10)
```

	user_id	item_id	rating	timestamp
0	0	50	5	881250949
1	0	172	5	881250949
2	0	133	1	881250949
3	196	242	3	881250949
4	186	302	3	891717742
5	22	377	1	878887116
6	244	51	2	880606923
7	166	346	1	886397596
8	298	474	4	884182806
9	115	265	2	881171488

```
movies_rating_df.tail()
```

	user_id	item_id	rating	timestamp
99998	880	476	3	880175444
99999	716	204	5	879795543
100000	276	1090	1	874795795
100001	13	225	2	882399156
100002	12	203	3	879959583

Figure 13: Second Dataset (Rating Dataset)

```
# Let's merge both dataframes together so we can have ID with the movie name
movies_rating_df = pd.merge(movies_rating_df, movie_titles_df, on = 'item_id')
```

```
movies_rating_df
```

	user_id	item_id	rating	title
0	0	50	5	Star Wars (1977)
1	290	50	5	Star Wars (1977)
2	79	50	4	Star Wars (1977)
3	2	50	5	Star Wars (1977)
4	8	50	5	Star Wars (1977)
5	274	50	5	Star Wars (1977)
6	227	50	4	Star Wars (1977)
7	99	50	5	Star Wars (1977)
8	305	50	5	Star Wars (1977)
9	108	50	4	Star Wars (1977)
10	63	50	4	Star Wars (1977)
11	234	50	4	Star Wars (1977)

Figure 14: Merging Dataset

STEP #2: VISUALIZE DATASET

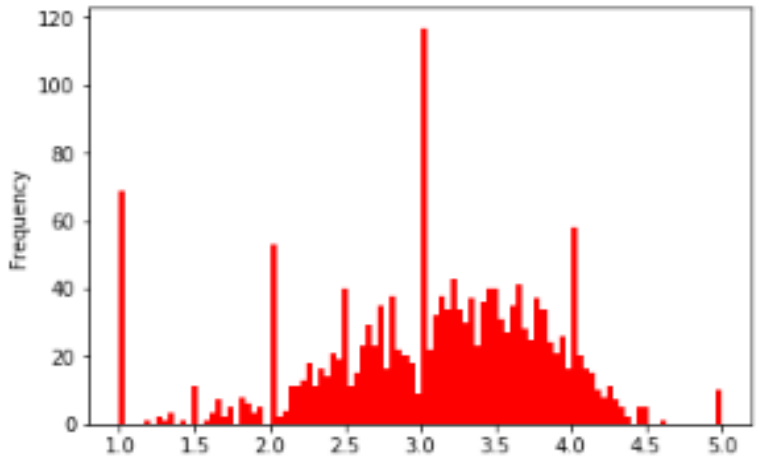
```
movies_rating_df.groupby('title')['rating'].describe()
```

title	count	mean	std	min	25%	50%	75%	max
'Til There Was You (1997)	9.0	2.333333	1.000000	1.0	2.00	2.0	3.00	4.0
1-900 (1994)	5.0	2.600000	1.516575	1.0	1.00	3.0	4.00	4.0
101 Dalmatians (1996)	109.0	2.908257	1.076184	1.0	2.00	3.0	4.00	5.0
12 Angry Men (1957)	125.0	4.344000	0.719588	2.0	4.00	4.0	5.00	5.0
187 (1997)	41.0	3.024390	1.172344	1.0	2.00	3.0	4.00	5.0
2 Days in the Valley (1996)	93.0	3.225806	0.957000	1.0	3.00	3.0	4.00	5.0
20,000 Leagues Under the Sea (1954)	72.0	3.500000	0.731581	2.0	3.00	4.0	4.00	5.0
2001: A Space Odyssey (1968)	259.0	3.969112	1.026307	1.0	3.00	4.0	5.00	5.0
3 Ninjas: High Noon At Mega Mountain (1998)	5.0	1.000000	0.000000	1.0	1.00	1.0	1.00	1.0
39 Steps, The (1935)	59.0	4.050847	0.859505	2.0	4.00	4.0	5.00	5.0
8 1/2 (1963)	38.0	3.815789	1.182194	1.0	3.00	4.0	5.00	5.0
8 Heads in a Duffel Bag (1997)	4.0	3.250000	1.500000	1.0	3.25	4.0	4.00	4.0
8 Seconds (1994)	4.0	3.750000	1.258306	2.0	3.50	4.0	4.25	5.0
A Chef in Love (1996)	8.0	4.125000	0.834523	3.0	3.75	4.0	5.00	5.0
Above the Rim (1994)	5.0	3.000000	0.707107	2.0	3.00	3.0	3.00	4.0
Absolute Power (1997)	127.0	3.370079	0.949546	1.0	3.00	3.0	4.00	5.0
Abyss, The (1989)	151.0	3.589404	0.911201	1.0	3.00	4.0	4.00	5.0
Ace Ventura: Pet Detective (1994)	103.0	3.048544	1.215735	1.0	2.00	3.0	4.00	5.0
Ace Ventura: When Nature Calls (1995)	37.0	2.675676	1.106899	1.0	2.00	3.0	3.00	5.0
Across the Sea of Time (1995)	4.0	2.750000	1.258306	1.0	2.50	3.0	3.25	4.0
Addams Family Values (1993)	87.0	2.816092	0.946582	1.0	2.00	3.0	3.00	5.0
Addicted to Love (1997)	54.0	3.166667	1.004706	1.0	3.00	3.0	4.00	5.0
Addiction, The (1995)	11.0	2.181818	1.328020	1.0	1.00	2.0	3.50	4.0
Adventures of Pinocchio, The (1996)	39.0	3.051282	1.145902	1.0	2.00	3.0	4.00	5.0
Adventures of Priscilla, Queen of the Desert, The (1994)	111.0	3.594595	1.030255	1.0	3.00	4.0	4.00	5.0
Adventures of Robin Hood, The (1938)	67.0	3.791045	0.844542	2.0	3.00	4.0	4.00	5.0
Affair to Remember, An (1957)	26.0	4.192308	0.895287	2.0	4.00	4.0	5.00	5.0

Figure 15: Visualize Dataset

```
ratings_mean_count_df['mean'].plot(bins=100, kind='hist', color = 'r')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x162279ab4e0>
```




```
ratings_mean_count_df['count'].plot(bins=100, kind='hist', color = 'r')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x16228172e80>
```

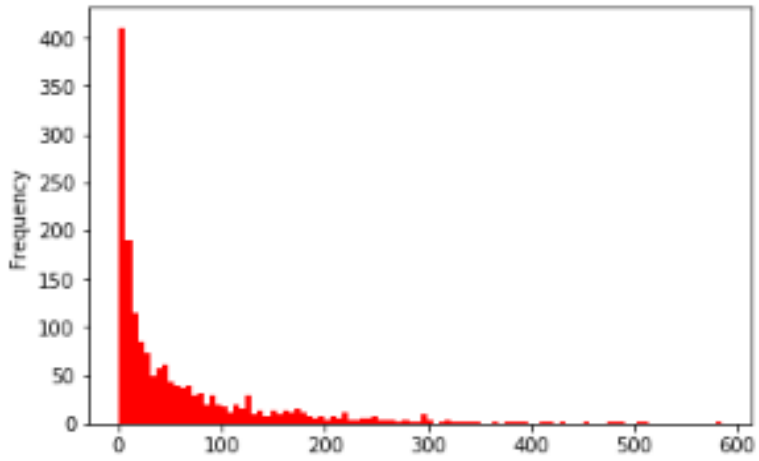


Figure 16: Dataset Graphs

STEP #3: PERFORM ITEM-BASED COLLABORATIVE FILTERING ON ONE MOVIE SAMPLE

```
userid_movietitle_matrix = movies_rating_df.pivot_table(index = 'user_id', columns = 'title', values = 'rating')
```

```
userid_movietitle_matrix
```

title	'Til There Was You (1997)	1-900 (1994)	101 Dalmatians (1996)	12 Angry Men (1957)	187 (1997)	2 Days in the Valley (1996)	20,000 Leagues Under the Sea (1954)	2001: A Space Odyssey (1968)	3 Ninjas: High Noon At Mega Mountain (1998)	39 Steps, The (1935)	...	Yankee Zulu (1994)	Year of the Horse (1997)	You So Crazy (1994)	Young Frankenstein (1974)	Young Guns (1988)	Young Guns II (1990)	P H T
user_id																		
0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	
1	NaN	NaN	2.0	5.0	NaN	NaN	3.0	4.0	NaN	NaN	...	NaN	NaN	NaN	5.0	3.0	NaN	
2	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	1.0	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	
3	NaN	NaN	NaN	NaN	2.0	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	
4	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	
5	NaN	NaN	2.0	NaN	NaN	NaN	NaN	4.0	NaN	NaN	...	NaN	NaN	NaN	4.0	NaN	NaN	
6	NaN	NaN	NaN	4.0	NaN	NaN	NaN	5.0	NaN	NaN	...	NaN	NaN	NaN	4.0	NaN	NaN	
7	NaN	NaN	NaN	4.0	NaN	NaN	5.0	5.0	NaN	4.0	...	NaN	NaN	NaN	5.0	3.0	NaN	

Figure 17: Item Based Filtering On One Movie

STEP#4: CREATE AN ITEM-BASED COLLABORATIVE FILTER ON THE ENTIRE DATASET

```
# Recall this matrix that we created earlier of all movies and their user ID/ratings
userid_movietitle_matrix
```

title	'Til There Was You (1997)	1-900 (1994)	101 Dalmatians (1996)	12 Angry Men (1957)	187 (1997)	2 Days in the Valley (1996)	20,000 Leagues Under the Sea (1954)	2001: A Space Odyssey (1968)	3 Ninjas: High Noon At Mega Mountain (1998)	39 Steps, The (1935)	...	Yankee Zulu (1994)	Year of the Horse (1997)	You So Crazy (1994)	Young Frankenstein (1974)	Young Guns (1988)	Young Guns II (1990)	P H T
user_id																		
0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	
1	NaN	NaN	2.0	5.0	NaN	NaN	3.0	4.0	NaN	NaN	...	NaN	NaN	NaN	5.0	3.0	NaN	
2	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	1.0	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	
3	NaN	NaN	NaN	NaN	2.0	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	
4	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	
5	NaN	NaN	2.0	NaN	NaN	NaN	NaN	4.0	NaN	NaN	...	NaN	NaN	NaN	4.0	NaN	NaN	
6	NaN	NaN	NaN	4.0	NaN	NaN	NaN	5.0	NaN	NaN	...	NaN	NaN	NaN	4.0	NaN	NaN	
7	NaN	NaN	NaN	4.0	NaN	NaN	5.0	5.0	NaN	4.0	...	NaN	NaN	NaN	5.0	3.0	NaN	

Figure 18: Item Based Filtering On Entire Dataset

```
# Let's create our own dataframe with our own ratings!
myRatings = pd.read_csv("My_Ratings.csv")
#myRatings.reset_index
```

```
myRatings
```

	Movie Name	Ratings
0	Liar Liar (1997)	5
1	Star Wars (1977)	1

```
len(myRatings.index)
```

```
2
```

```
myRatings['Movie Name'][0]
```

```
'Liar Liar (1997)'
```

```
similar_movies_list = pd.Series()
for i in range(0, 2):
    similar_movie = movie_correlations[myRatings['Movie Name'][i]].dropna() # Get same movies with same ratings
    similar_movie = similar_movie.map(lambda x: x * myRatings['Ratings'][i]) # Scale the similarity by your given ratings
    similar_movies_list = similar_movies_list.append(similar_movie)
```

```
similar_movies_list.sort_values(inplace = True, ascending = False)
print (similar_movies_list.head(10))
```

```
Liar Liar (1997)                          5.000000
Con Air (1997)                            2.349141
Pretty Woman (1990)                       2.348951
Michael (1996)                            2.210110
Indiana Jones and the Last Crusade (1989)  2.072136
Top Gun (1986)                            2.028602
G.I. Jane (1997)                          1.989656
Multiplicity (1996)                       1.984302
Grumpier Old Men (1995)                   1.953494
Ghost and the Darkness, The (1996)        1.895376
dtype: float64
```

EXCELLENT JOB! NOW YOU BECAME FAMILIAR WITH RECOMMENDER SYSTEMS, GREAT JOB!

Figure 19: Find Similar Movie

Chapter 7

Conclusion & Future Scope

Recommender systems open new opportunities of retrieving personalized information on the Internet. It also helps to alleviate the problem of information overload which is a very common phenomenon with information retrieval systems and enables users to have access to products and services which are not readily available to users on the system. We come up with a strategy that focuses on dealing with user's personal interests and based on his previous reviews, movies are recommended to users. This strategy helps in improving accuracy of the recommendations. A personal profile is created for each user, where each user has access to his own history, his likes, ratings, comments, password modification processes. It also helps in collecting authentic data with improved accuracy and makes the system more responsive.

A hybrid approach is taken between context based filtering and collaborative filtering to implement the system. This approach overcomes drawbacks of each individual algorithm and improves the performance of the system. Techniques like Clustering, Similarity and Classification are used to get better recommendations thus reducing MAE and increasing precision and accuracy.

In future we can work on hybrid recommender using clustering and similarity for better performance. Our approach can be further extended to other domains to recommend songs, video, venue, news, books, tourism and e-commerce sites, etc.

REFERENCES

1. Peterson, Benjamin (20 April 2020). "Python Insider: Python 2.7.18, the last release of Python " *Python Insider*. Retrieved 27 April 2020.

2. *Automated Design of Both the Topology and Sizing of Analog Electrical Circuits Using Genetic Programming. Artificial Intelligence in Design '96. Springer, Dordrecht. pp. 151–170. doi:10.1007/978-94-009-0279-4_9.*

3. Bishop, C. M. (2006), *Pattern Recognition and Machine Learning*, Springer, ISBN 978-0-387-31073-2

4. Machine learning and pattern recognition "can be viewed as two facets of the same field.

5. Schutt, Rachel; O'Neil, Cathy (2013). *Doing Data Science*. O'Reilly Media. ISBN 978-1-449-35865-5.

6. "Data Cleaning". Microsoft Research. Retrieved 26 October 2013.

7. Perceptual Edge-Jonathan Koomey-Best practices for understanding quantitative data-February 14, 2006

8. Hellerstein, Joseph (27 February 2008). "Quantitative Data Cleaning for Large Databases" (PDF). *EECS Computer Science Division*: 3. Retrieved 26 October 2013.

09. Stephen Few-Perceptual Edge-Selecting the Right Graph For Your Message-September 2004

10. Behrens-Principles and Procedures of Exploratory Data Analysis-American Psychological Association-1997

11. Kuhlman, Dave. "A Python Book: Beginning Python, Advanced Python, and Python Exercises". Section 1.1. Archived from the original (PDF) on 23 June 2012.

12. https://grouplens.org/datasets/movielens/100k/

13. Kuhlman, Dave. "A Python Book: Beginning Python, Advanced Python, and Python Exercises". Section 1.1. Archived from the original (PDF) on 23 June 2012.

14. *About Python". Python Software Foundation. Retrieved 24 April 2012.*, second section "Fans of Python use the phrase "batteries included" to describe the standard library, which covers everything from asynchronous processing to zip files.

15. Peterson, Benjamin (20 April 2020). "Python Insider: Python 2.7.18, the last release of Python 2". *Python Insider*. Retrieved 27 April 2020.

16. *"Sunsetting Python 2". Python.org. Retrieved 22 September 2019.*

17. "PEP 373 -- Python 2.7 Release Schedule". *Python.org*. Retrieved 22 September 2019.

18. *"History and License". Retrieved 5 December 2016.* "All Python releases are Open Source"

19. Venners, Bill (13 January 2003). "The Making of Python". *Artima Developer*. Artima. Retrieved 22 March 2007.

20. van Rossum, Guido (29 August 2000). "SETL (was: Lukewarm about range literals)". *Python-Dev* (Mailing list). Retrieved 13 March 2011.

21. van Rossum, Guido (20 January 2009). "A Brief Timeline of Python". *The History of Python*. Retrieved 20 January 2009.

22. Fairchild, Carlie (12 July 2018). "Guido van Rossum Stepping Down from Role as Python's Benevolent Dictator For Life". *Linux Journal*. Retrieved 13 July 2018.

23. "Guido van Rossum Stepping Down from Role as Python's Benevolent Dictator For Life | Linux Journal". *www.linuxjournal.com*.

24. "Python boss Guido van Rossum steps down after 30 years". *The Inquirer*.

25. "PEP 8100". Python Software Foundation. Retrieved 4 May 2019.

26. "Is Python a good language for beginning programmers?". *General Python FAQ*. Python Software Foundation. Retrieved 21 March 2007.

27. "Myths about indentation in Python". Secnetix.de. Retrieved 19 April 2011.

28. "The Notebook, Qt console and a number of other pieces are now parts of Jupyter".

29. sshirokov (2015-05-07). "GitHub + Jupyter Notebooks = <3". *The GitHub Blog*. Retrieved 2018-04-10.

30. "Rendering Notebooks on GitHub – Jupyter Blog". *Jupyter Blog*. 2015-05-07. Retrieved 2018-04-10.

31. Books referred:

31.1 “Machine Learning Yearning” by Andrew Ng **Publication** Year: 2018

31.2“Artificial Intelligence in Practice” by Bernard Marr **Publication** Year: 2016

32. Web sites referred:

32.1 www.python.org

32.2 http://ijesc.org/

32.3 www.datapine.com

32.4 online.visual-paradigm.com

32.5 www.movielens.com

32.6 www.github.com

33. Peng, Xiao, Shao Liangshan, and Li Xiuran. "Improved Collaborative Filtering Algorithm in the Research and Application of Personalized Movie Recommendations", 2013 Fourth International Conference on Intelligent Systems Design and Engineering Applications, 2013.

34. Munoz-Organero, Mario, Gustavo A. Ramíez-González, Pedro J. Munoz-Merino, and Carlos Delgado Kloos. "A Collaborative Recommender System Based on Space-Time Similarities", IEEE Pervasive Computing, 2010.

35. Al-Shamri, M.Y.H.. "Fuzzy-genetic approach to recommender systems based on a novel hybrid user model", Expert Systems With Applications, 200810

36. Hu Jinming. "Application and research of collaborative filtering in e-commerce recommendation system", 2010 3rd International Conference on Computer Science and Information Technology, 07/2010

37. Xu, Qingzhen Wu, Jiayong Chen, Qiang. "A novel mobile personalized recommended method based on money flow model for stock exchange.(Researc", Mathematical Problems in Engineering, Annual 2014 Issue

38. Yan, Bo, and Guanling Chen. "AppJoy : personalized mobile application discovery", Proceedings of the 9th international conference on Mobile systems applications and services - MobiSys 11 MobiSys 11, 2011.