PERFORMANCE COMPARISON OF RECOMMENDER SYSTEMS

PROJECT – I REPORT

Submitted by

REG NO	NAME
61072011130	NITHISHKUMAR S
61072011125	MAHAKASILINGAM A

Submitted in partial fulfilment for award of the degree

of

BACHELOR OF ENGINEERING

In

COMPUTER SCIENCE AND ENGINEERING

Under the guidance of

Dr. S.Selvi M.E., Ph.D.,

Assistant professor (Sr. Gr)

GOVERNMENT COLLEGE OF ENGINEERING BARGUR,
KRISNAGIRI-635 104.
(Affiliated to Anna University, Accredited by NAAC with 'B' Grade)

ANNA UNIVERSITY:CHENNAI 600 025

DECEMBER 2022

ANNA UNIVERSITY: CHENNAI 600 025

BONAFIDE CERTIFICATE

Certified that this project REPORT titled

"**PERFORMANCE COMPARISON OF RECOMMENDER SYSTEMS**"

is the bonafide work of NITHISHKUMAR S(61072011130), MAHAKASILINGAM A(61072011125) who carried out the mini project work under my supervision.

Dr. S.Selvi M.E., Ph.D.,
SUPERVISOR
Assistant Professor (Sr. Gr),
Department of CSE,
Govt. College of Engineering,
Bargur-635 104.

Dr.J.NAFEESA BEGUM, M.E.,Ph.D.,
HEAD OF THE DEPARTMENT
Professor,
Department of CSE,
Govt. College of Engineering,
Bargur-635 104.

Submitted for the Mini Project Viva Voce Examination held on _____ at GOVERNMENT COLLEGE OF ENGINEERING BARGUR, KRISHNAGIRI- 635 104.

INTERNAL EXAMINER **EXTERNAL EXAMINER**

ACKNOWLEDGEMENT

We express our sincere of gratitude to our Principal **Dr. R.Vijayan, M.E., PhD**, for his timely advice and his encouragement.

We express our regards to **Dr. J.NAFEESA BEGUM M.E.,Ph.D., Head of Department** for her support, coordinate and for providing necessary facilities whenever needed to carry out the project.

We consider it as a great privilege to place a record or our sense of gratitude to our Internal Guide **Dr.S.SELVI PhD.,** Assistant professor, Department of Computer Science and Engineering for beginning the great inspiration and motivation to us to do the project, consistently guided us in completing the project.

At the very outset, we wish to express our sincere thanks to all those who were more involved in the completion of this project.

Our most sincere salutations go to Anna University, Chennai that gave us an opportunity to have sound base of Computer Science and Engineering.

We also express our thanks to our department faculty members, our parents and our beloved friends.

ABSTRACT

Performance Comparison of Recommender Algorithms

The recommender systems are used in many product-based companies for profiling users based on the content they generate. The data that gets generated include personalised information and user interaction data. These data are then evaluated to generate the personalised content for a user. The user may get a nice user experience when using the software application implemented with recommendation system.

The recommender system model implements four various algorithms for performance comparison. It requires large dataset for training and evaluation. After acquiring the datasets pre-processing begins. The pre-processing includes cleaning, instance selection, normalization, transformation, feature extraction. Missing values are nullified in cleaning process, instance selection involves selection of required data items. The data items are normalized produce a new instance which are transformed and a new feature is extracted. After pre-processing training of dataset begins. In training process, all features are analysed statistically and the model is evaluated. One key aspect of evaluation is to ensure that the trained model generalizes for data it was not trained on, using Cross-validation techniques. As there are different instance types, we associate them with a weight or strength, assuming that, for example, a comment in an article indicates a higher interest of the user on the item than a like, or than a simple view. This process is called data munging. Data munging of four different algorithms produces four different weights or strength values. The four different values are compared in a pictorial representation for better visualization.

The common way to assess the performance of a recommender system would be through standard metrics such as Accuracy, Precision or Recall. However, these metrics require ground truth knowledge about which recommendations are correct. The best recommendation algorithm is then chosen and is tested for appropriate user information.

LIST OF FIGURES

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

INTRODUCTION

1.1. What is machine learning?

Machine Learning is the science (and art) of programming computers so they can learn from data. Here is a slightly more general definition:

"Machine Learning is the field of study that gives computers the ability to learn without being explicitly programmed"

– said by Arthur Samuel, 1959. And a more engineering-oriented one: A computer program is said to learn from experience E with respect to some task T and some performance measure P, if its performance on T, as measured by P, improves with experience E. – said by Tom Mitchell, 1997. For example, your spam filter is a Machine Learning program that can learn to flag spam given examples of spam emails (e.g., flagged by users) and examples of regular (non - spam, also called "ham") emails.

1.2. Why use machine learning?

1. What would look at spam typically looks like. May notice that some words or phrases (such as "4U," "credit card," "free," and "amazing") tend to come up a lot in the subject. Perhaps you would also notice a few other patterns in the sender's name, the email's body, and so on.

2. You would write a detection algorithm for each of the patterns that you noticed, and your program would flag emails as spam if a number of these patterns are detected.

3. You would test your program, and repeat steps 1 and 2 until it is good enough

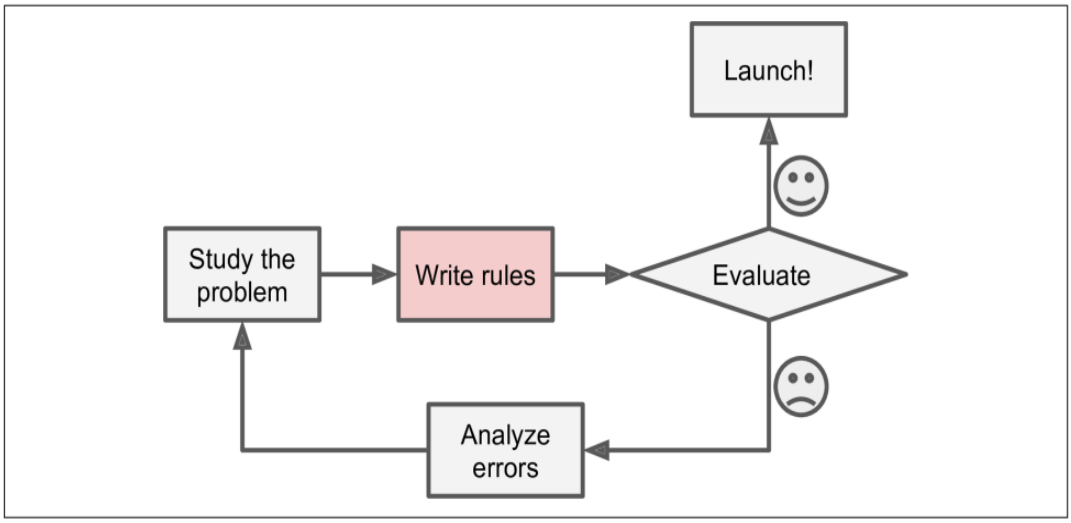


Figure 1.1 – Traditional Approach

In contrast, a spam filter based on Machine Learning techniques automatically learns which words and phrases are good predictors of spam by detecting unusually frequent patterns of words in the spam examples compared to the ham examples (Figure 1-2). The program is much shorter, easier to maintain, and most likely more accurate.

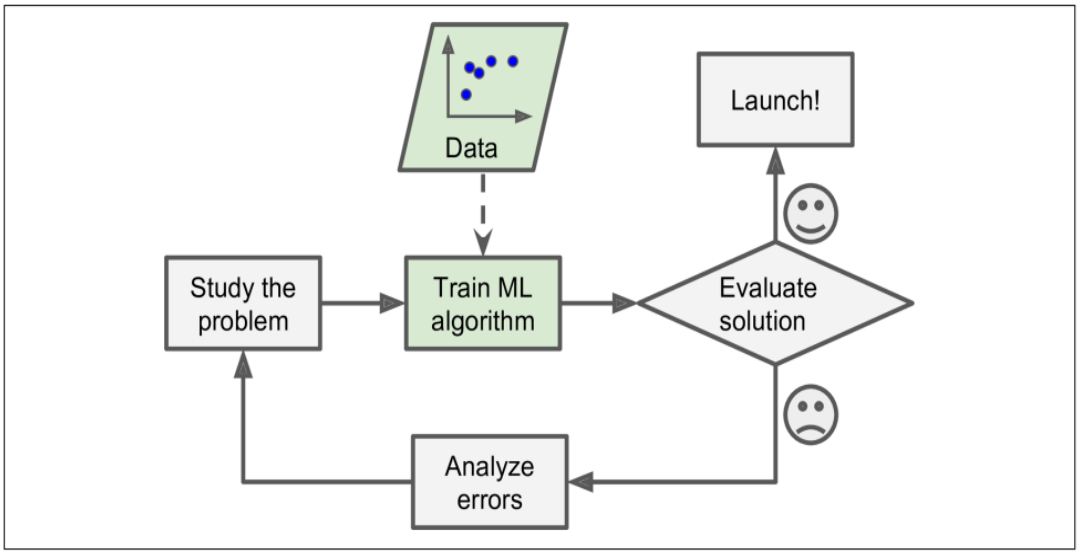


Figure 1.2 – Machine learning approach

Moreover, if spammers notice that all their emails containing “4U” are blocked, they might start writing “For U” instead. A spam filter using traditional programming techniques would need to be updated to flag “For U” emails. If spammers keep working around your spam filter, you will need to keep writing new rules forever. In contrast, a spam filter based on Machine Learning techniques automatically notices that “For U” has become unusually frequent in spam flagged by users, and it starts flagging them without your intervention (Figure 1-3).

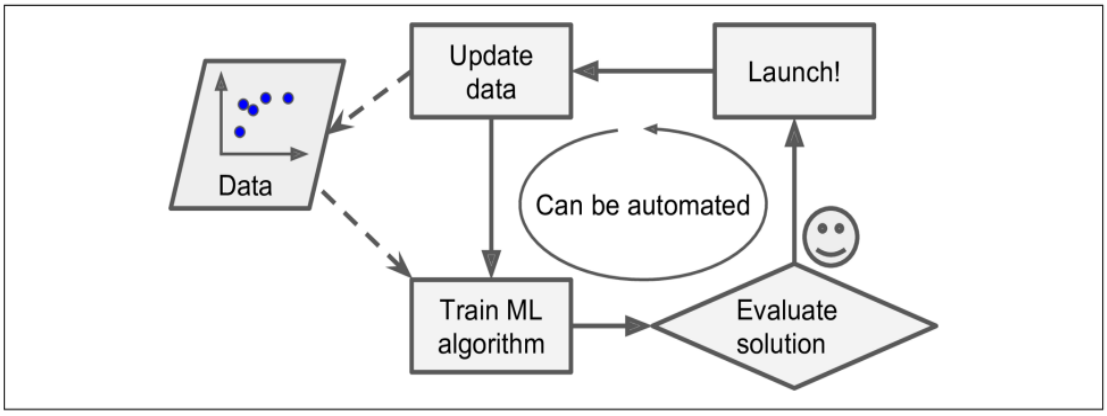


Figure 1.3 – Automatically adapting to change

1.3. Types of machine learning:

Machine Learning systems can be classified according to the amount and type of supervision they get during training. There are four major categories: supervised learning, unsupervised learning, semi supervised learning, and Reinforcement Learning.

Supervised Learning

In supervised learning, the training data you feed to the algorithm includes the desired solutions, called labels (Figure 1-5).

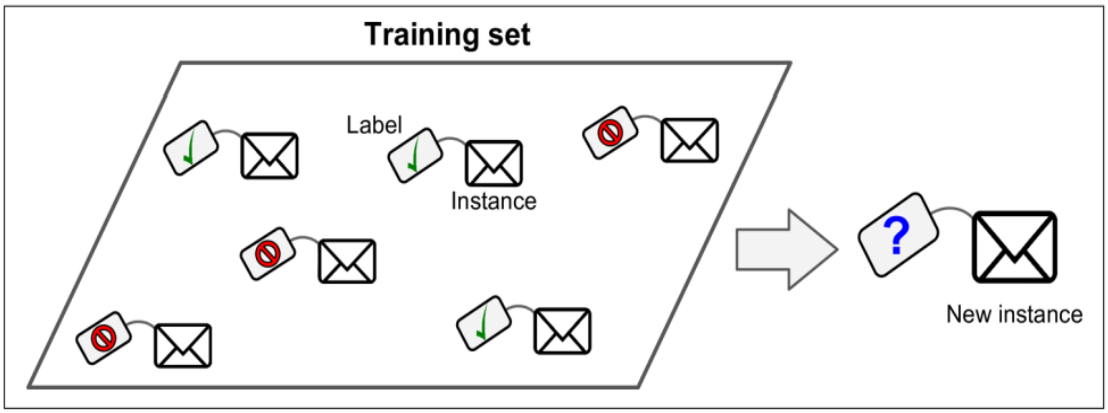


Figure 1.4 – Training set for supervised learning

Unsupervised Learning

In unsupervised learning, as you might guess, the training data is unlabelled (Figure 1-7). The system tries to learn without a teacher.

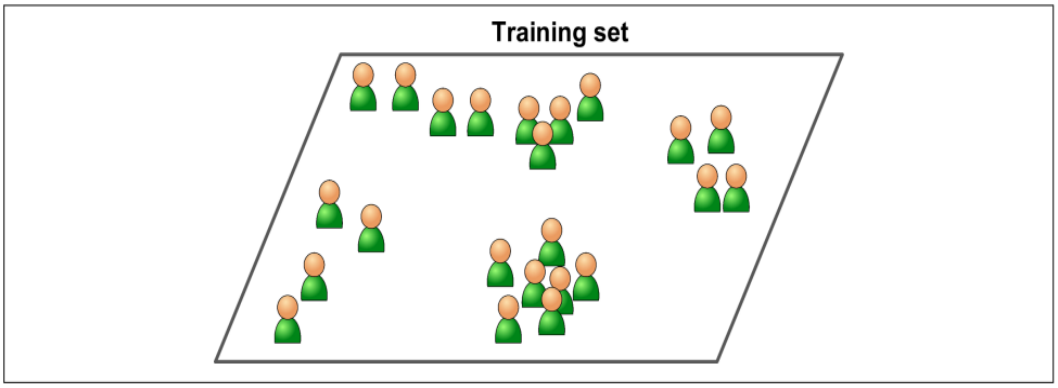


Figure 1.5 – Training set for unsupervised learning

Reinforcement Learning

Reinforcement Learning is a very different beast. The learning system, called an agent in this context, can observe the environment, select and perform actions, and get rewards in return (or penalties in the form of negative rewards, as in Figure 1-12). It must then learn by itself what is the best strategy, called a policy, to get the most reward over time. A policy defines what action the agent should choose when it is in a given situation.

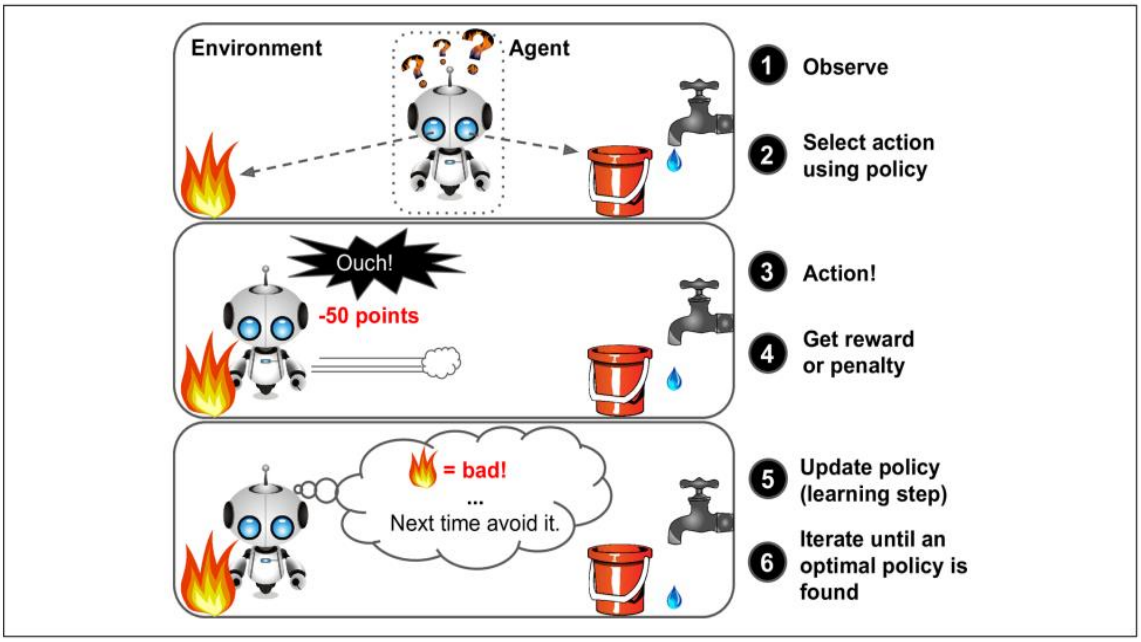


Figure 1.6 – Training set for Reinforcement learning

1.4. Recommendation Algorithms

The objective of a recommendation system is to recommend relevant items for users, based on their preference. Preference and relevance are subjective, and they are generally inferred by items users have consumed previously.

- **Collaborative Filtering**: This method makes automatic predictions (filtering) about the interests of a user by collecting preferences or taste information from many users (collaborating). The underlying assumption of the collaborative filtering approach is that if a person A has the same opinion as a person B on a set of items, A is more likely to have B's opinion for a given item than that of a randomly chosen person.
- **Content-Based Filtering**: This method uses only information about the description and attributes of the items users has previously consumed to model user's preferences. In other words, these algorithms try to recommend items that are similar to those that a user liked in the past (or is examining in the present). In particular, various candidate items are compared with items previously rated by the user and the best-matching items are recommended.
- **Hybrid methods**: Recent research has demonstrated that a hybrid approach, combining collaborative filtering and content-based filtering could be more effective than pure

approaches in some cases. These methods can also be used to overcome some of the common problems in recommender systems such as cold start and the sparsity problem.

- **Popularity**: It is a type of recommendation system which works on the principle of popularity and or anything which is in trend. These systems check about the articles or blogs which are in trend or are most popular among the users and directly recommend.

It is proposed to load the Deskdrop dataset which contains a real sample of 12 months logs (Mar. 2016 - Feb. 2017) from CI&T's Internal Communication platform (DeskDrop) for training. Also, it contains about 73k logged user's interactions on more than 3k public articles shared in the platform. It is composed of two CSV files namely shared_articles.csv, users_interactions.csv. A model will be constructed by the training dataset. Finally the constructed model tests the remaining data for evaluate the performance of the models.

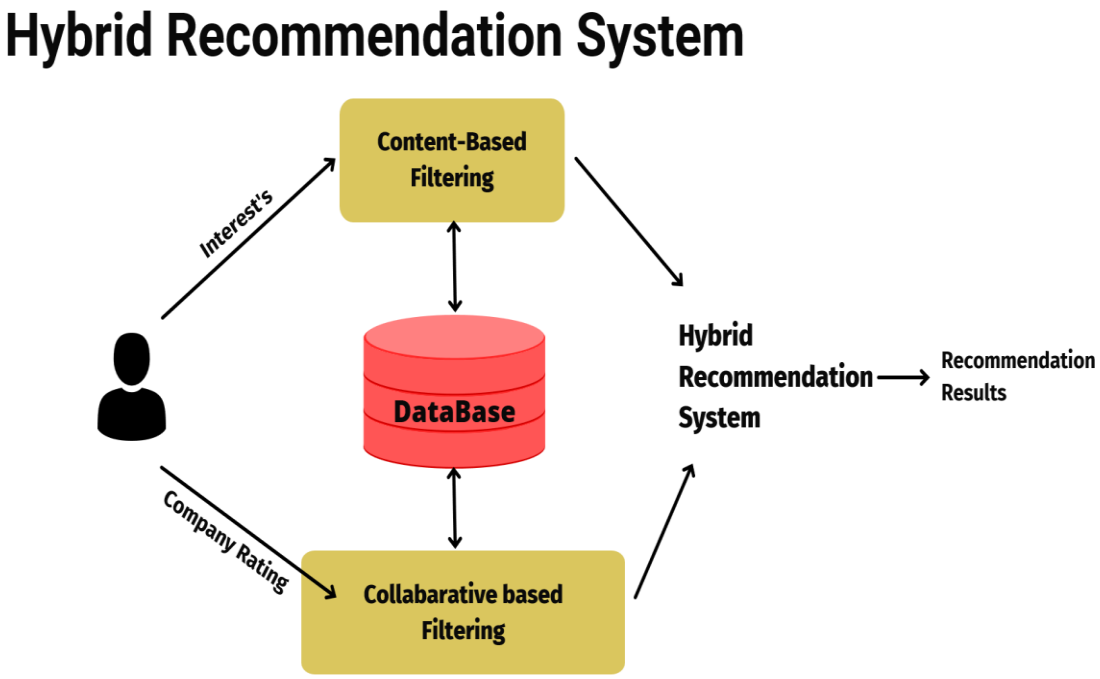


Figure 1.7 – Hybrid recommender system

1.5 Project description

There is a growing demand for recommendation systems in many tech companies. As the data on the internet grows, there is a need to recommend users with their preferences becomes must. The companies collect vital data from their users and use it for analysing their preference. Therefore, the companies need to provide contents based on their preference. There are many learning algorithms of which three algorithms stand out from the rest. They are content based filtering, collaborative based filtering, and hybrid recommendation model. Thus the performance analysis between these models become extremely necessary the analysis among these algorithms is done by training with large data set and interpreting the result is a human readable form ie in graph. Thus the objective also include

a) Generation of large dataset
b) Training the dataset with these three algorithms
c) Interpreting the result in a graph

CHAPTER 2

SYSTEM ANALYSIS

SYSTEM ANALYSIS

2.1 LITERATURE SURVEY

2.1.1 Content-based recommendation system

The research paper published in November 2008 titled "Content based recommendation system" by four authors namely Charilaos Zisopoulus, Savvas Karagiannidis, Georgios Demirtsoglou and Stefanos Antaris. The paper contained the basic idea behind the content based recommendation system. It was started that it is used towards web technologies. As more and more businesses are pushed towards the web. The contents of the web increases exponentially. The users also expect a nice user experience with the website they are web browsing.

The idea that was stated in the paper was "The items are recommended to the user based on the item's description and the preference of the user." Thus, the item on a website must have a discrete, identifiable and unique description. The user's data are to be collected based on some more unique descriptions.

2.1.2 Collaborative filtering recommender system

The research paper released in January, 2007 titled "Collaborative filtering recommender system" by authors Ben Schafer, Ben J, Dan Frankowski. The idea proposed by then was very simple. It is based on the opinion given by other users. The author Ben Schafer even cites an example of a movie, that is if a user wants to watch a movie, the user can also about other user's opinion and decide whether to watch the movie or not. The sharing of opinion is dated long back way before the advent of computer technologies.

The opinions of the users are used to describe an item and given a particular value. When the user might be interested in that item, it gets recommended. More robust mathematical model is discovered and the collaborative filtering method is getting better every year.

2.1.3 Hybrid recommender systems: A systematic literature review

The content based and collaborative model have their own pros and cons. Some researchers even went on research of combining these two approaches. Some of the research paper include “Hybrid recommender system: A systematic literature review” by Erino Cano and Maurizio Morisio in November 2017. The best features of the two model are combined to promote the algorithm to the next level. The classical idea of content based is used with filtering to get a unique description of an item, then the item gets served to the user with a particular preference.

2.2 EXISTING SYSTEM

The algorithms that have been proposed had many shortcomings. The comparison algorithms after the data have been trained into the system. This algorithm has a flaw that it does not know about how the data has been trained on by the recommendation system. The other algorithms defined are based on the output of the each of the recommendation algorithms. They just always compare the results of individual algorithm.

2.2.1 Disadvantages

- They are plain, just facts about the result of the recommendation systems.
- Does not know about how models train on data.
- Takes up so much space
- Complexity increases as the number of recommendation algorithms to compare increase.
- Does not always yield good result.

2.3 PROPOSED SYSTEM

The proposed system consists of an algorithm based on the machine learning principles to compare a variety of recommendation system. The proposed algorithm takes care of how the data gets trained and evaluated to give the best. The results are very intuitive as it gets interpreted in a human readable friendly form, i.e in a graph. The algorithm also checks the results for a wide variety of input dataset.

2.3.1 Advantages

- Deals with the details of how the data gets trained.
- Results are interpreted in a nice form.
- Give good comparison results even on different large and different datasets.

CHAPTER 3

SYSTEM REQUIREMENTS

SYSTEM REQUIREMENTS

3.1 HARDWARE REQUIREMENTS:

The hardware requirements may serve as the basis for a contract for the implementation of the system and should therefore be a complete and consistent specification of the whole system. They are used by software engineers as the starting point for the system design. It should what the system does and not how it should be implemented.

RAM: 8 GB

GPU: AMD Radeon 4 GB

Processor: AMD Ryzen 5

Hard disk: 512 GB

3.2 SOFTWARE REQUIREMENTS:

The software requirements document is the specification of the system. It should include both a definition and a specification of requirements. It is a set of what the system should do rather than how it should do it. The software requirements provide a basis for creating the software requirements specification. It is useful in estimating cost, planning team activities, performing tasks and tracking the teams and tracking the team's progress throughout the development.

Operating System: Windows 11

Programming Language: Python

Processing Software: Jupyter Noteboook

Python Core Modules: Pandas, NumPy, Sklearn, NLTK, SciPy

3.3 SOFTWARE DESCRIPTION:

3.3.1 Jupyter Notebook

The Jupyter Notebook App is a server-client application that allows editing and running notebook documents via a web browser. The Jupyter Notebook App can be executed on a local desktop requiring no internet access (as described in this document) or can be installed on a remote server and accessed through the internet.

In addition to displaying/editing/running notebook documents, the Jupyter Notebook App has a "Dashboard" (Notebook Dashboard), a "control panel" showing local files and allowing to open notebook documents or shutting down their kernels.

3.3.2 Pandas

Pandas is an open-source Python Library providing high-performance data manipulation and analysis tool using its powerful data structures. The name Pandas is derived from the word Panel Data – an Econometrics from Multidimensional data.

In 2008, developer Wes McKinney started developing pandas when in need of high performance, flexible tool for analysis of data.

Prior to Pandas, Python was majorly used for data munging and preparation. It had very little contribution towards data analysis. Pandas solved this problem. Using Pandas, we can accomplish five typical steps in the processing and analysis of data, regardless of the origin of data — load, prepare, manipulate, model, and analyze.

Python with Pandas is used in a wide range of fields including academic and commercial domains including finance, economics, Statistics, analytics, etc.

Key features:

- Fast and efficient DataFrame object with default and customized indexing.
- Tools for loading data into in-memory data objects from different file formats.
- Data alignment and integrated handling of missing data.

- Reshaping and pivoting of date sets.
- Label-based slicing, indexing and subsetting of large data sets.
- Columns from a data structure can be deleted or inserted.
- Group by data for aggregation and transformations.
- High performance merging and joining of data.
- Time Series functionality.

3.3.3 NumPy

NumPy is a Python package. It stands for 'Numerical Python'. It is a library consisting of multidimensional array objects and a collection of routines for processing of array.

Numeric, the ancestor of NumPy, was developed by Jim Hugunin. Another package Numarray was also developed, having some additional functionalities. In 2005, Travis Oliphant created NumPy package by incorporating the features of Numarray into Numeric package. There are many contributors to this open-source project.

Operations on NumPy

Using NumPy, a developer can perform the following operations −

- Mathematical and logical operations on arrays.
- Fourier transforms and routines for shape manipulation.
- Operations related to linear algebra. NumPy has in-built functions for linear algebra and random number generation.

3.3.4 SkLearn

Scikit-learn (Sklearn) is the most useful and robust library for machine learning in Python. It provides a selection of efficient tools for machine learning and statistical modelling including classification, regression, clustering, and dimensionality reduction via a consistence interface in Python. This library, which is largely written in Python, is built upon NumPy, SciPy and Matplotlib.

Features

Rather than focusing on loading, manipulating, and summarizing data, Scikit-learn library is focused on modeling the data. Some of the most popular groups of models provided by Sklearn are as follows –

Supervised Learning algorithms − Almost all the popular supervised learning algorithms, like Linear Regression, Support Vector Machine (SVM), Decision Tree etc., are the part of scikit-learn.

Unsupervised Learning algorithms − On the other hand, it also has all the popular unsupervised learning algorithms from clustering, factor analysis, PCA (Principal Component Analysis) to unsupervised neural networks.

Clustering − This model is used for grouping unlabeled data.

Cross Validation − It is used to check the accuracy of supervised models on unseen data.

Dimensionality Reduction − It is used for reducing the number of attributes in data which can be further used for summarization, visualization, and feature selection.

Ensemble methods − As name suggest, it is used for combining the predictions of multiple supervised models.

Feature extraction − It is used to extract the features from data to define the attributes in image and text data.

Feature selection − It is used to identify useful attributes to create supervised models.

Open Source − It is open-source library and also commercially usable under BSD license.

3.3.5 NLTK

Language is a method of communication with the help of which we can speak, read and write. Natural Language Processing (NLP) is the sub field of computer science especially Artificial Intelligence (AI) that is concerned about enabling computers to understand and process human language. We have various open-source NLP tools but NLTK (Natural Language Toolkit) scores very high when it comes to the ease of use and explanation of the concept. The learning curve of Python is very fast and NLTK is written in Python so NLTK is also having very good learning kit. NLTK has incorporated most of the tasks like tokenization,

stemming, Lemmatization, Punctuation, Character Count, and Word count. It is very elegant and easy to work with.

3.3.6 SciPy

SciPy is a scientific python open source, distributed under the BSD licensed library to perform Mathematical, Scientific and Engineering Computations.

The SciPy library depends on NumPy, which provides convenient and fast N-dimensional array manipulation. The SciPy library is built to work with NumPy arrays and provides many user-friendly and efficient numerical practices such as routines for numerical integration and optimization. Together, they run on all popular operating systems, are quick to install and are free of charge. NumPy and SciPy are easy to use, but powerful enough to depend on by some of the world's leading scientists and engineers.

CHAPTER 4

SYSTEM DESIGN

4.1 SYSTEM ARCHITECTURE

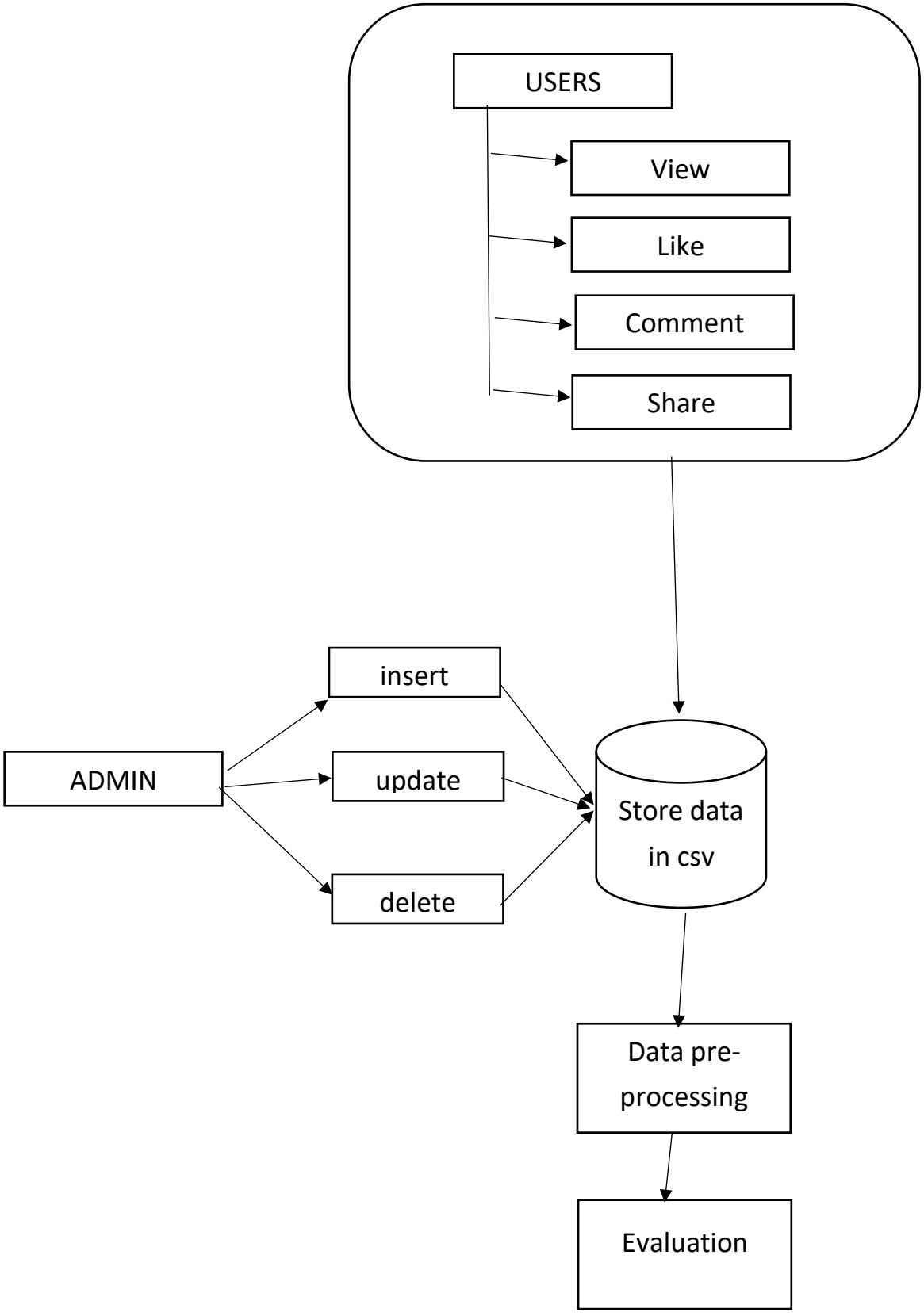


Figure 4.1 – System Architecture

The various data generated by the users include likes, shares, views and comments. These are generated at the client side of the application and store in a database server in .csv format in backend. This database serves as a dataset for the data pre-processing. The data is given as input to the algorithms. These data can be modified by the system administrator. These operations performed by the administrator include insert, update, delete. Then the data is given to algorithms to train. After training the outcomes of the algorithms are compared by a comparison algorithm are the result is evaluated. The evaluated result is interpreted in a nice graph.

4.2 DATAFLOW DIAGRAM

4.2.1 Level 0 - Data flow Diagram for hybrid recommender system

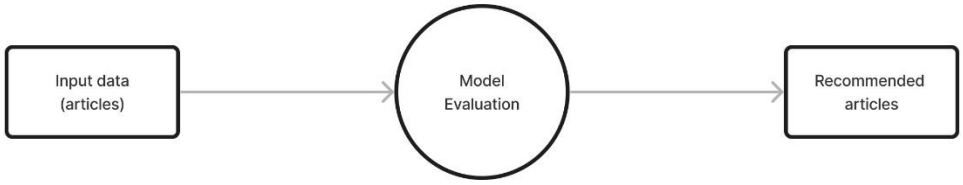


Figure 4.2 – Dataflow diagram level 0

The input data consists of likes, view, comments and share data about various articles, for example. It is given to the hybrid recommender system for evaluation. The result of the various algorithm will be of recommended articles based on the users' interests.

4.2.2 Level 1 - Data flow Diagram for hybrid recommender system

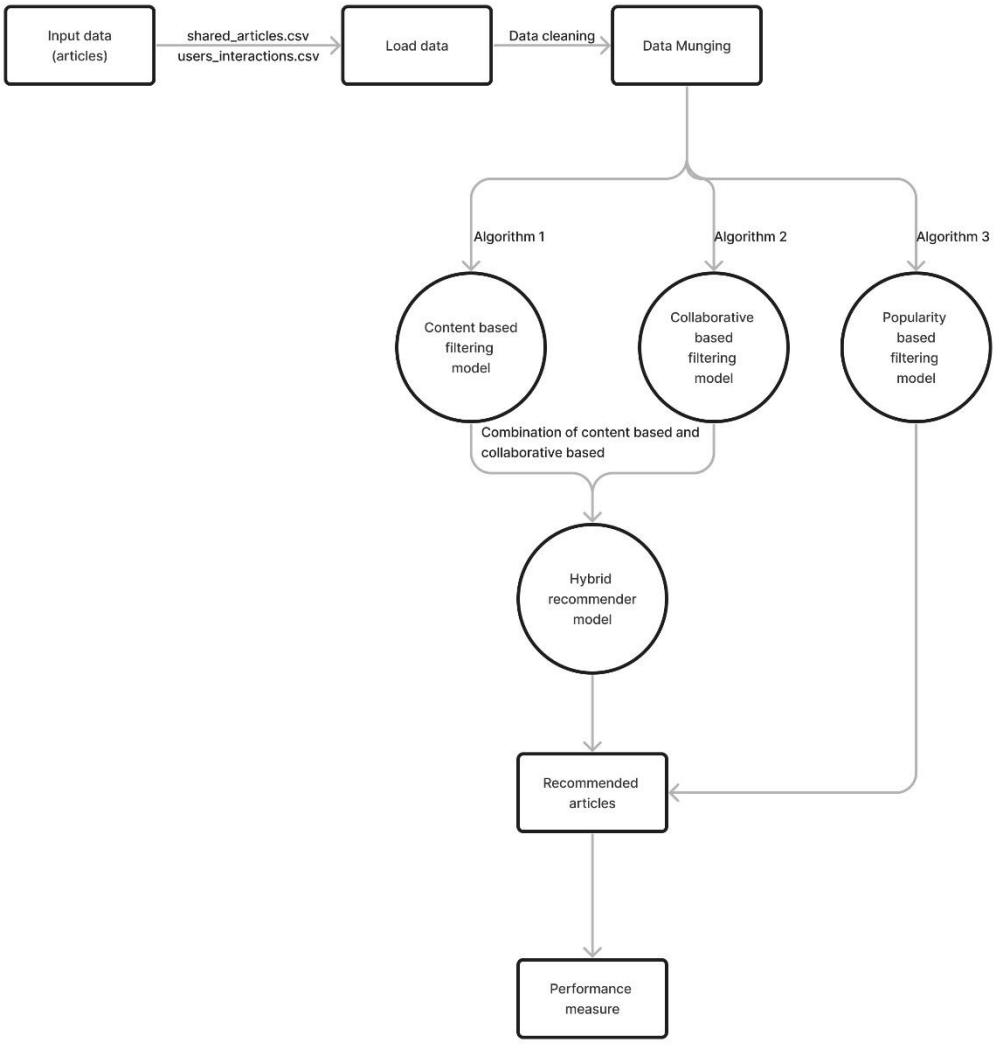


Figure 4.3 – Dataflow diagram level 1

The input data more accurately, consists of the articles in a file called shared-articles.csv and the user's interactions are collected and put in a file called user_interaction.csv. These are datasets required. Before feeding this data to the algorithms the loading of this data is required.

The loading phase consists of cleaning of data is done. The data cleaning is very essentially becoming some redundant can cause a performance change in the outcome of the algorithms. The datasets are then given to the algorithms. The algorithms are content based filtering, collaborative based filtering, and popularity-based filtering. The outcome of the first two algorithms is then combined to give hybrid recommendation algorithm. These outcomes are analysed by using various performance measures ore metrics.

4.2.3 Level 2 - Data flow Diagram for hybrid recommender system

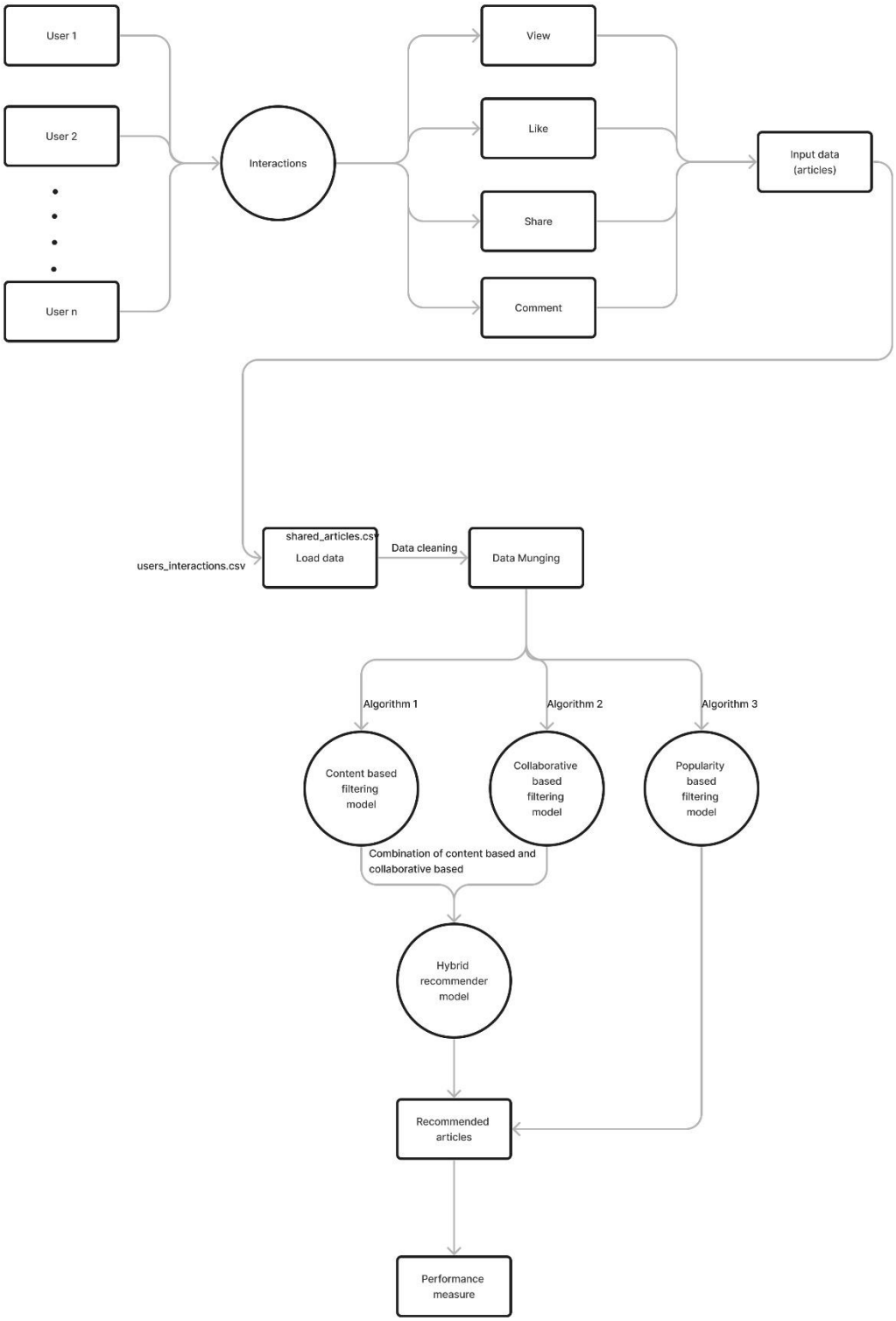


Figure 4.4 - Dataflow diagram level 2

A recommender system, or a recommendation system, can be thought of as a subclass of information filtering system that seeks to predict the best “rating” or “preference” a user would give to an item which is typically obtained by optimizing for objectives like total clicks, total revenue, and overall sales. Broadly speaking, most recommender systems leverage two types of data namely, interaction Data, such as ratings, and browsing behaviours, and attribution information, about each user and items.

The modelling approach relying on the former data is generally known Collaborative Filtering method, and the approach using the latter is referred to as the Content-Base Filtering method. There is also another category known as Knowledge-Based recommender system that is based on explicitly specified user requirements. Of course, each of these methods has its strengths and weaknesses depending on which applications they are used for, and the amount of data available. Hybrid Systems are then used to combined the advantages of these approaches to have a robust performing system across a wide variety of applications.

4.3 UML DIAGRAM

The elements are like components which can be associated in different ways to make complete UML pictures which is known as diagram. So, it is very important to understand the different diagrams to implement the knowledge in real life systems.

We prepare UML diagrams to understand a system in better and simple way. A single diagram is not enough to cover all aspects of the system. So, UML defines various kinds of diagrams to cover most of the aspects of a system.

4.3.1 Use case diagram

A use case illustrates a unit of functionality provided by the system. The main purpose of the use-case diagram is to help development teams visualize the functional requirements of a system, including the relationship of "actors" (human beings who will interact with the system) to essential processes, as well as the relationships among different use cases. Use-case diagrams generally show groups of use cases -- either all use cases for the complete system, or a breakout of a particular group of use cases with related functionality.

Use Case Diagram are behaviour diagrams used to describe a set of actions (use-cases) that some system or systems (subject) should or can perform in collaboration with one or more external users of the system (actor). Here we have a system like administrator actions like as rating, view, like, share, comment to the article.

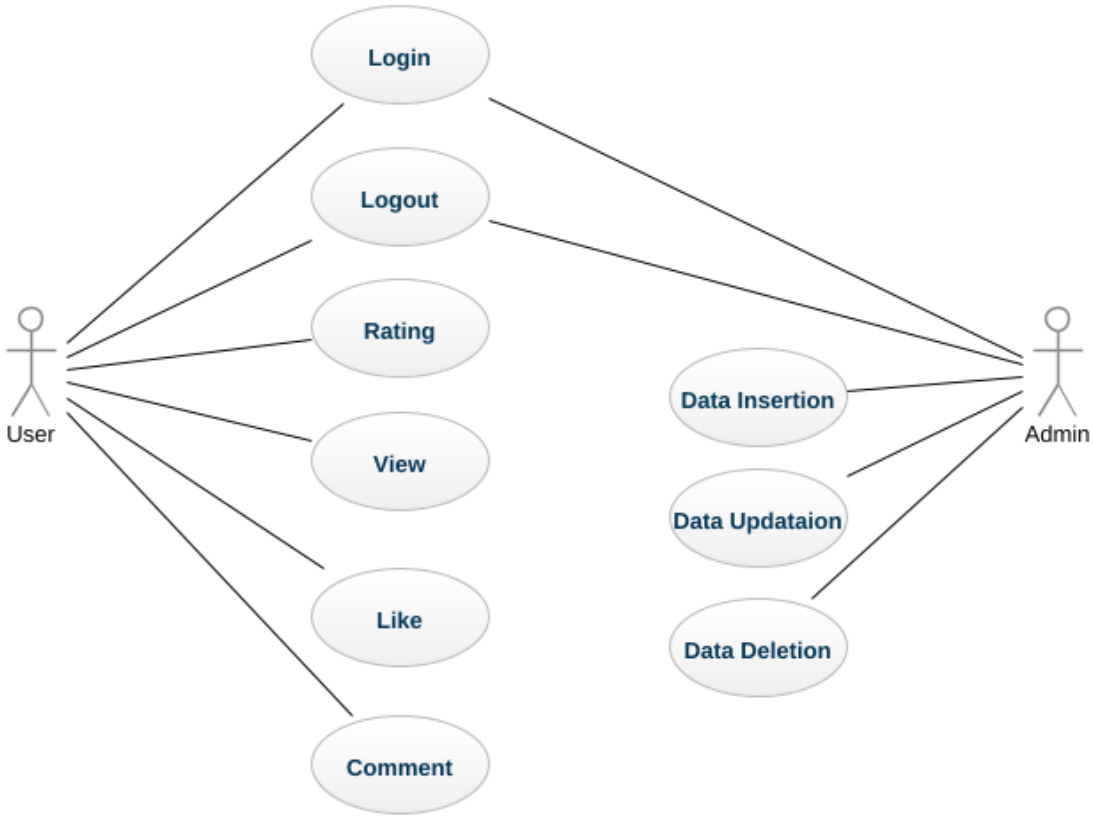


Figure 4.5 – Use case diagram for recommendation system

4.3.2 Activity Diagram

Activity diagrams show the procedural flow of control between two or more class objects while processing an activity. Activity diagrams can be used to model higher-level business process at the business unit level, or to model low-level internal class actions. An activity diagram's notation set is like that used in a state chart diagram. Like a state chart diagram, the activity diagram starts with a solid circle connected to the initial activity

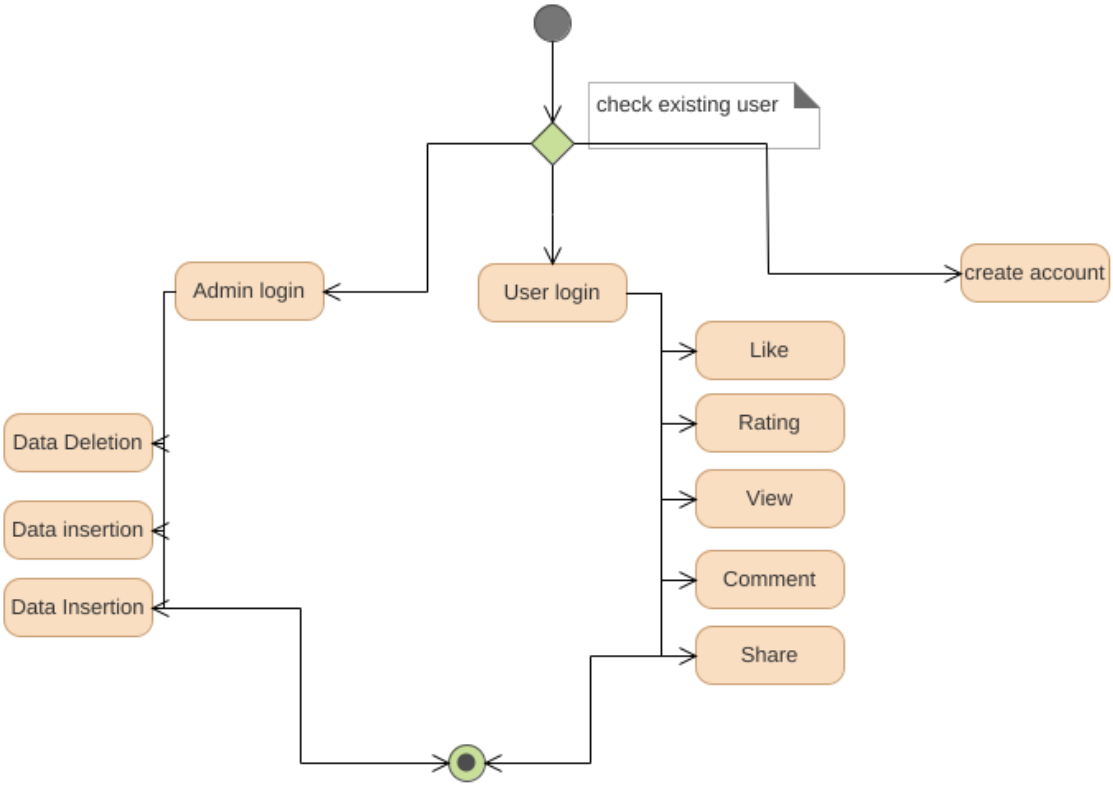


Figure 4.6 – Activity diagram for recommendation system

4.3.3 Sequence diagram

Sequence diagrams show a detailed flow for a specific use case or even just part of a specific use case. They are almost self-explanatory; they show the calls between the different objects in their sequence and can show, at a detailed level, different calls to different objects. A sequence diagram has two dimensions: The vertical dimension shows the sequence of messages/calls in the time order that they occur, the horizontal dimension shows the object instances to which the messages are sent.

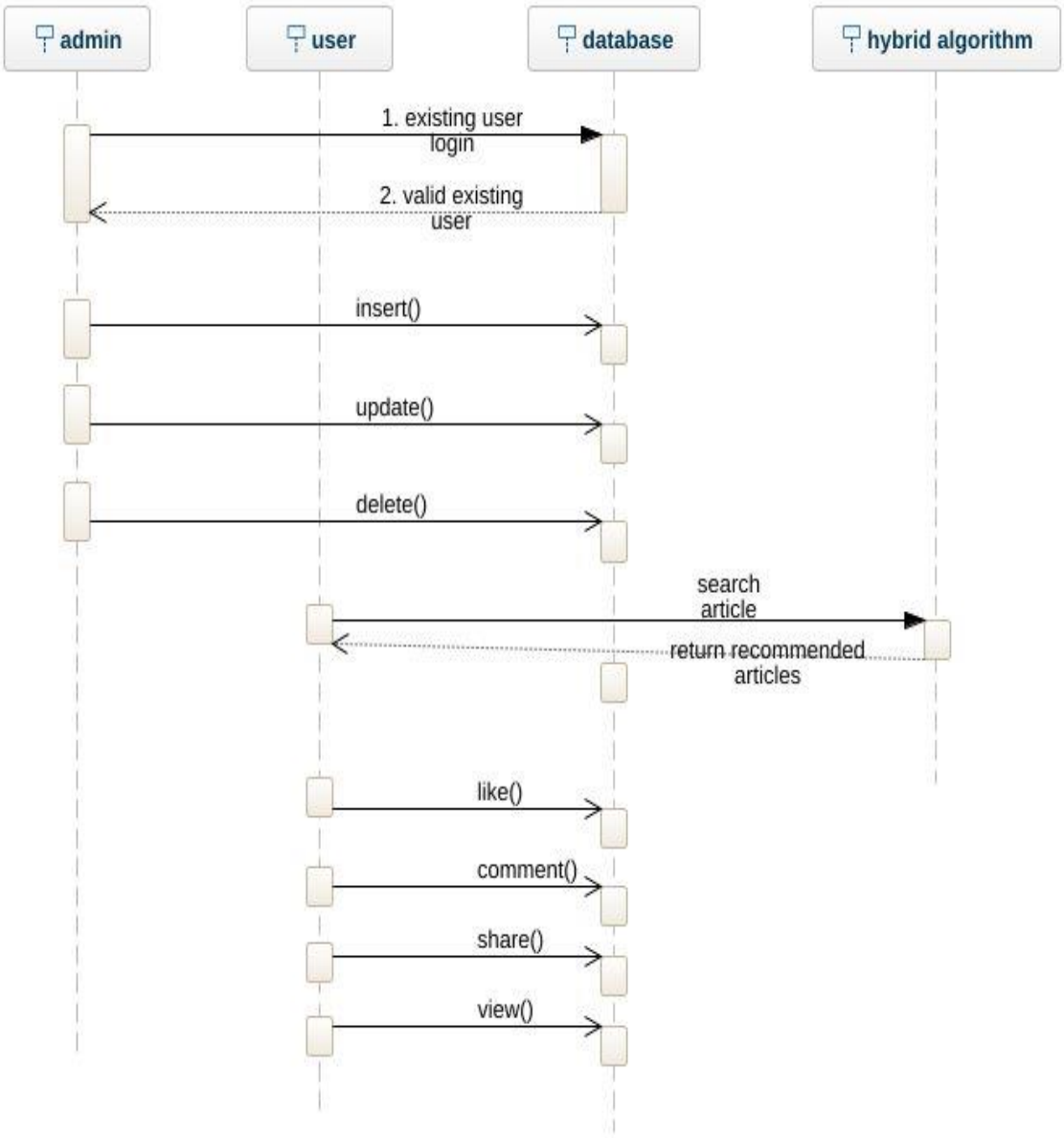


Figure 4.7 - Sequence diagram for hybrid recommendation system

CHAPTER 5

MODULES DESCRIPTION

5.1 MODULES

The proposed system contains the following modules

- Data pre-processing
- Training data
- Evaluate for Contend based
- Evaluate for Collaborative filtering
- Evaluate for popularity based
- Evaluate for hybrid recommender system

Class diagram

The class diagram shows how the different entities (people, things, and data) relate to each other, in other words, it shows the static structures of the system. A class diagram can be used to display logical classes, which are typically the kinds of things the business people in an organization talk about -- rock bands, CDs, radio play, or loans, home mortgages, car loans, and interest rates. Class diagrams can also be used to show implementation classes, which are the things that programmers typically deal with an implementation class diagram will probably show some of the same classes as the logical class diagram. Class Diagram provides an overview of the target system by describing the objects and classes inside the system and the relationships between them. It provides a wide variety of usages from modelling the domain-specific data structure to detailed design of the target system. Each class have a class name with attributes and operations Suppose admin have a one class means user name and password is the attributes, login, insert, update, delete are the operations of class diagram.

Class diagram provides an overview of the target system by describing the objects and classes.

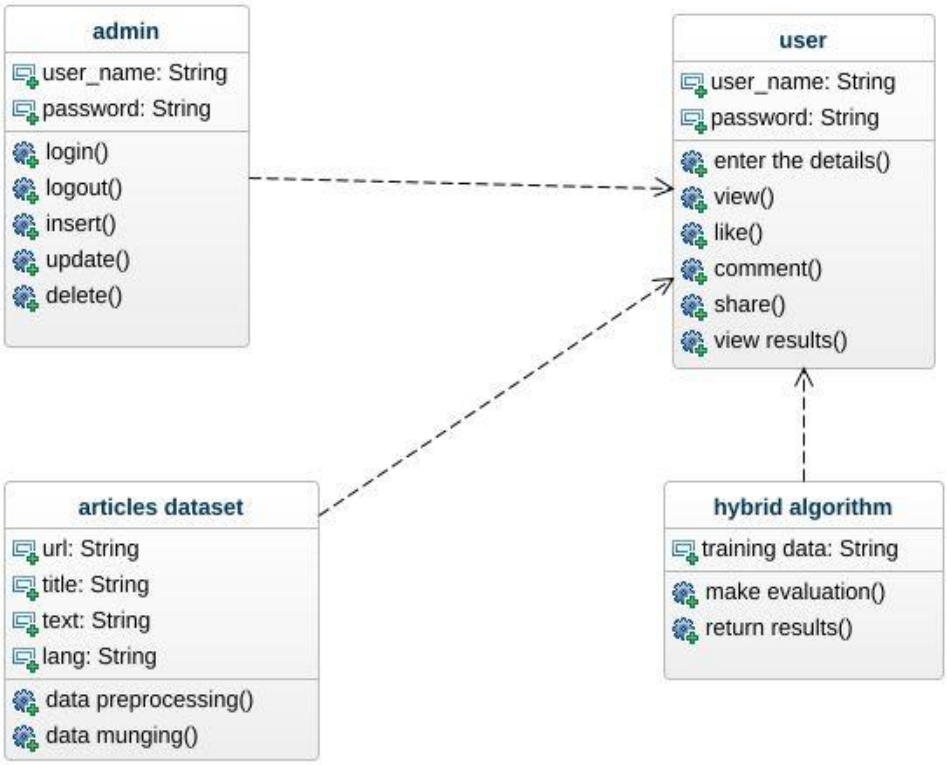


Fig 5.1 Class diagram for hybrid recommender system

5.1.1 Data pre-processing

Data pre-processing is a process of preparing the raw data and making it suitable for a machine learning model. It is the first and crucial step while creating a machine learning model. When creating a machine learning project, it is not always a case that we come across the clean and formatted data. And while doing any operation with data, it is mandatory to clean it and put in a formatted way. So for this, we use data pre-processing task. A real-world data generally contains noises, missing values, and maybe in an unusable format which cannot be directly used for machine learning models. Data pre-processing is required tasks for cleaning the data and making it suitable for a machine learning model which also increases the accuracy and efficiency of a machine learning model.


```
users_interactions_count_df =
interactions_df.groupby(['personId','contentId']).size().groupby('personId').size()

print('# users: %d' % len(users_interactions_count_df))

users_with_enough_interactions_df =
users_interactions_count_df[users_interactions_count_df >= 5].reset_index()[['personId']]

print('# users with at least 5 interactions: %d' % len(users_with_enough_interactions_df))
```

5.1.2 Training Data

In our project, we load the Deskdrop dataset, which contains a real sample of 12 months logs (Mar. 2016 - Feb. 2017) from CI&T's Internal Communication platform (DeskDrop). It contains about 73k logged users' interactions on more than 3k public articles shared in the platform. It is composed of two CSV files:

- shared_articles.csv
- users_interactions.csv

shared_articles.csv

Contains information about the articles shared in the platform. Each article has its sharing date (timestamp), the original URL, title, content in plain text, the article' lang (Portuguese: pt or English: en) and information about the user who shared the article (author).

There are two possible event types at a given timestamp:

CONTENT SHARED: The article was shared in the platform and is available for users.

CONTENT REMOVED: The article was removed from the platform and not available for further recommendation.

For the sake of simplicity, we only consider here the "CONTENT SHARED" event type, assuming (naively) that all articles were available during the whole one year period. For a more precise evaluation (and higher accuracy), only articles that were available at a given time should be recommended.

```
articles_df = pd.read_csv('../input/shared_articles.csv')
articles_df = articles_df[articles_df['eventType'] == 'CONTENT SHARED']
articles_df.head(5)
```

User_interactions.csv

Contains logs of user interactions on shared articles. It can be joined to articles_shared.csv by contentId column.

The eventType values are:

- VIEW: The user has opened the article.
- LIKE: The user has liked the article.
- COMMENT CREATED: The user created a comment in the article.
- FOLLOW: The user chose to be notified on any new comment in the article.
- BOOKMARK: The user has bookmarked the article for easy return in the future.

```
interactions_df = pd.read_csv('../input/users_interactions.csv')
interactions_df.head(10)
```

5.1.3 Evaluate for Content based filtering

Content-based filtering approaches leverage description or attributes from items the user has interacted to recommend similar items. It depends only on the user previous choices, making this method robust to avoid the cold-start problem. For textual items, like articles, news and books, it is simple to use the raw text to build item profiles and user profiles.

Here we are using a very popular technique in information retrieval (search engines) named TF-IDF. This technique converts unstructured text into a vector structure, where each word is represented by a position in the vector, and the value measures how relevant a given word is for an article. As all items will be represented in the same Vector Space Model, it is to compute similarity between articles.

5.1.4 Evaluate for Collaborative based filtering

Collaborative Filtering (CF) has two main implementation strategies:

Memory-based: This approach uses the memory of previous users' interactions to compute users' similarities based on items they've interacted (user-based approach) or compute items similarities based on the users that have interacted with them (item-based approach).

A typical example of this approach is User Neighbourhood-based CF, in which the top-N similar users (usually computed using Pearson correlation) for a user are selected and used to recommend items those similar users liked, but the current user have not interacted yet. This approach is very simple to implement, but usually do not scale well for many users. A nice Python implementation of this approach in available in Crab.

Model-based: This approach, models are developed using different machine learning algorithms to recommend items to users. There are many model-based CF algorithms, like neural networks, Bayesian networks, clustering models, and latent factor models such as Singular Value Decomposition (SVD) and, probabilistic latent semantic analysis.

Matrix factorization

Latent factor models compress user-item matrix into a low-dimensional representation in terms of latent factors. One advantage of using this approach is that instead of having a high dimensional matrix containing abundant number of missing values we will be dealing with a much smaller matrix in lower-dimensional space.

A reduced presentation could be utilized for either user-based or item-based neighbourhood algorithms that are presented in the previous section. There are several advantages with this paradigm. It handles the sparsity of the original matrix better than memory-based ones. Also comparing similarity on the resulting matrix is much more scalable especially in dealing with large sparse datasets.

Here we a use popular latent factor model named Singular Value Decomposition (SVD). There are other matrix factorization frameworks more specific to CF, like surprise, mrec or python-recsys. We chose a SciPy implementation of SVD because it is available on kernels.

An important decision is the number of factors to factor the user-item matrix. The higher the number of factors, the more precise is the factorization in the original matrix reconstructions. Therefore, if the model is allowed to memorize too much details of the original matrix, it may not generalize well for data it was not trained on. Reducing the number of factors increases the model generalization.

class **CFRecommender**:

```
    MODEL_NAME = 'Collaborative Filtering'

    def __init__(self, cf_predictions_df, items_df=None):
        self.cf_predictions_df = cf_predictions_df
        self.items_df = items_df

    def get_model_name(self):
        return self.MODEL_NAME

    def recommend_items(self, user_id, items_to_ignore=[], topn=10, verbose=False):

        sorted_user_predictions = self.cf_predictions_df[user_id].sort_values(ascending=False)\
                                .reset_index().rename(columns={user_id: 'recStrength'})

        recommendations_df = sorted_user_predictions[~sorted_user_predictions['contentId'].isi
n(items_to_ignore)] \ .sort_values('recStrength', ascending = False) \ .head(topn)

        if verbose:
            if self.items_df is None:
                raise Exception('"items_df" is required in verbose mode')

            recommendations_df = recommendations_df.merge(self.items_df, how = 'left',
                                left_on = 'contentId', right_on = 'contentId')[['recStrength', 'contentId',
                                'title', 'url', 'lang']]
            return recommendations_df

cf_recommender_model = CFRecommender(cf_preds_df, articles_df)
```

5.1.5 Evaluate for popularity-based filtering

A common (and usually hard-to-beat) baseline approach is the Popularity model. This model is not actually personalized - it simply recommends to a user the most popular items that the user has not previously consumed. As the popularity accounts for the "wisdom of the crowds", it usually provides good recommendations, generally interesting for most people.

The main objective of a recommender system is to leverage the long-tail items to the users with very specific interests, which goes far beyond this simple technique.

```
class PopularityRecommender:

    MODEL_NAME = 'Popularity'

    def __init__(self, popularity_df, items_df=None):
        self.popularity_df = popularity_df
        self.items_df = items_df

    def get_model_name(self):
        return self.MODEL_NAME

    def recommend_items(self, user_id, items_to_ignore=[], topn=10, verbose=False):
        recommendations_df = self.popularity_df[~self.popularity_df['contentId'].isin(items_to_
ignore)] \.sort_values('eventStrength', ascending = False) \ .head(topn)

        if verbose:
            if self.items_df is None:
                raise Exception('"items_df" is required in verbose mode')

            recommendations_df = recommendations_df.merge(self.items_df, how = 'left',
                                                          left_on = 'contentId',
                                                          right_on = 'contentId')[['eventStrength', 'contentId', 'title',
'url', 'lang']]


        return recommendations_df

popularity_model = PopularityRecommender(item_popularity_df, articles_df)
```

5.1.6 Evaluate for hybrid recommender system

What if we combine Collaborative Filtering and Content-Based Filtering approaches?

Would that provide us with more accurate recommendations?

In fact, hybrid methods have performed better than individual approaches in many studies and have been extensively used by researchers and practioners.

Let us build a simple hybridization method, as an ensemble that takes the weighted average of the normalized CF scores with the Content-Based scores, and ranking by resulting score. In this case, as the CF model is much more accurate than the CB model, the weights for the CF and CB models are 100.0 and 1.0, respectively.

```
class HybridRecommender:

    MODEL_NAME = 'Hybrid'

    def __init__(self, cb_rec_model, cf_rec_model, items_df, cb_ensemble_weight=1.0, cf_en
semble_weight=1.0):
        self.cb_rec_model = cb_rec_model
        self.cf_rec_model = cf_rec_model
        self.cb_ensemble_weight = cb_ensemble_weight
        self.cf_ensemble_weight = cf_ensemble_weight
        self.items_df = items_df

    def get_model_name(self):
        return self.MODEL_NAME

    def recommend_items(self, user_id, items_to_ignore=[], topn=10, verbose=False):
        cb_recs_df = self.cb_rec_model.recommend_items(user_id, items_to_ignore=items_to_i
gnore, verbose=verbose,topn=1000).rename(columns={'recStrength': 'recStrengthCB'})

        cf_recs_df = self.cf_rec_model.recommend_items(user_id, items_to_ignore=items_to_i
gnore, verbose=verbose, topn=1000).rename(columns={'recStrength': 'recStrengthCF'})

        recs_df = cb_recs_df.merge(cf_recs_df,
                                   how = 'outer',
                                   left_on = 'contentId',
                                   right_on = 'contentId').fillna(0.0)

        recs_df['recStrengthHybrid'] = (recs_df['recStrengthCB'] * self.cb_ensemble_weight) \
                                       + (recs_df['recStrengthCF'] * self.cf_ensemble_weight)
```

```
        recommendations_df = recs_df.sort_values('recStrengthHybrid', ascending=False).head(
topn)

        if verbose:
            if self.items_df is None:
                raise Exception('"items_df" is required in verbose mode')

            recommendations_df = recommendations_df.merge(self.items_df, how = 'left',
                                                          left_on = 'contentId',
                                                          right_on = 'contentId')[['recStrengthHybrid', 'contentId', 't
itle', 'url', 'lang']]


        return recommendations_df

hybrid_recommender_model = HybridRecommender(content_based_recommender_model, c
f_recommender_model, articles_df, cb_ensemble_weight=1.0, cf_ensemble_weight=100.0)
```

CHAPTER 6

SYSTEM TESTING

6.1 OBJECTIVES OF TESTING

Testing is a process of executing a program with the intent of finding an error. A good test case is one that has a high probability of finding an as yet undiscovered error.

A successful test is one that uncovers a yet undiscovered error.

6.2 TESTING STRATEGIES

The extent of testing a system is controlled by many factors, such as the risk involved, limitations on resources, and deadlines. There are many testing strategies, but most used testing combinations are as follows:

- White Box Testing
- Black Box Testing
- Top-Down Testing
- Bottom-Up Testing

6.2.1 White Box Testing

In white box testing the specific logic is important and must be tested to guarantee the system's proper functioning. The main use of white box testing is in error-based testing.

6.2.2 Black Box Testing

The concept of black box is used to represent a system whose inside workings are not available for inspection. In Black box testing the logic is unknown; all that is known is what goes in and what comes out, or the input or output.

6.2.3 Top-Down Testing

In Top-Down testing the main logic or object interactions and system messages of the application need more testing than an individual object's methods or supporting logic.

6.2.4 Bottom-Up Testing

Bottom-Up testing starts with the details of the system and proceeds to higher levels by a progressive aggregation of details until they collectively fit the requirements for the system. This approach is more appropriate for testing the individual objects in a system.

6.2.5 Unit Testing

Unit testing concentrates on each unit of the software as implemented in source code. Unit testing makes heavy use of white box testing techniques, exercising specific path in a module's control structure to ensure complete coverage and maximum error detection.

6.2.6 Integration Testing

Integration testing focuses on design and the construction of the software architecture. Integration testing addressed the issues associated with the dual problems of verification and program construction. Developing a strategy for integrating the various web pages of a web site into functioning whole requires careful planning.

6.3 TESTING RESULTS

	eventStrength	Content id	title	url	lang
115	4.285402	7342707578347442862	At eBay, Machine Learning is Driving Innovativ...	https://www.ebayinc.com/stories/news/at-ebay-m...	en
38	4.129283	621816023396605502	AI Is Here to Help You Write Emails People Wil...	http://www.wired.com/2016/08/boomerang-using-a...	en
8	4.044394	-4460374799273064357	Deep Learning for Chatbots, Part 1 - Introduction	http://www.wildml.com/2016/04/deep-learning-fo...	en
116	3.954196	-7959318068735027467	Auto-scaling scikit-learn with Spark	https://databricks.com/blog/2016/02/08/auto-sc...	en
10	3.906891	2589533162305407436	6 reasons why I like KeystoneML	http://radar.oreilly.com/2015/07/6-reasons-why...	en
28	3.700440	5258604889412591249	Machine Learning Is No Longer Just for Experts	https://hbr.org/2016/10/machine-learning-is-no...	en
6	3.700440	-398780385766545248	10 Stats About Artificial Intelligence That Wi...	http://www.fool.com/investing/2016/06/19/10-st...	en
113	3.643856	-6467708104873171151	5 reasons your employees aren't sharing their ...	http://justcuriousblog.com/2016/04/5-reasons-y...	en

42	3.523562	-4944551138301474550	Algorithms and architecture for job recommenda...	https://www.oreilly.com/ideas/algorithms-and-a...	en
43	3.459432	-8377626164558006982	Bad Writing Is Destroying Your Company's Produ...	https://hbr.org/2016/09/bad-writing-is-destroy...	en
41	3.459432	444378495316508239	How to choose algorithms for Microsoft Azure M...	https://azure.microsoft.com/en-us/documentatio...	en
3	3.321928	2468005329717107277	How Netflix does A/B Testing - uxdesign.cc - U...	https://uxdesign.cc/how-netflix-does-a-b-testi...	en
101	3.321928	-8085935119790093311	Graph Capabilities with the Elastic Stack	https://www.elastic.co/webinars/sneak-peek-of-...	en
107	3.169925	-1429167743746492970	Building with Watson Technical Web Series	https://www-304.ibm.com/partnerworld/wps/servl...	pt
16	3.169925	6340108943344143104	Text summarization with TensorFlow	https://research.googleblog.com/2016/08/text-s...	en
49	3.169925	1525777409079968377	Probabilistic Programming	http://probabilistic-programming.org/wiki/Home	en
44	3.169925	-5756697018315640725	Being A Developer After 40 - Free Code Camp	https://medium.freecodecamp.com/being-a-develo...	en

97	3.087463	2623290164732957912	Creative Applications of Deep Learning with Te...	https://www.kadenze.com/courses/creative-appli...	en
32	3.000000	279771472506428952	5 Unique Features Of Google Compute Engine Tha...	http://www.forbes.com/sites/janakirammsv/2016/...	en
78	2.906891	-3920124114454832425	Worldwide Ops in Minutes with DataStax & Cloud	http://www.datastax.com/2016/01/datastax-enter...	en

	recStrengthHybrid	ContentId	title	url	lang
0	25.436876	3269302169678465882	The barbell effect of machine learning.	http://techcrunch.com/2016/06/02/the-barbell-e...	en
1	25.369932	-8085935119790093311	Graph Capabilities with the Elastic Stack	https://www.elastic.co/webinars/sneak-peek-of-...	en
2	24.493428	1005751836898964351	Seria Stranger Things uma obra de arte do algo...	https://www.linkedin.com/pulse/seria-stranger-...	pt
3	24.382997	-8377626164558006982	Bad Writing Is Destroying Your Company's Produ...	https://hbr.org/2016/09/bad-writing-is-destroy...	en
4	24.362064	-6727357771678896471	This Super Accurate Portrait Selection Tech Us...	http://petapixel.com/2016/06/29/super-accurate...	en
5	24.190327	-8190931845319543363	Machine Learning Is At The Very Peak Of Its Hy...	https://arc.applause.com/2016/08/17/gartner-hy...	en
6	24.172285	7395435905985567130	The AI business landscape	https://www.oreilly.com/ideas/the-ai-business-...	en
7	23.932289	5092635400707338872	Power to the People: How One Unknown Group of ...	https://medium.com/@atduskgreg/power-to-the-pe...	en
8	23.865716	-5253644367331262405	Hello, TensorFlow!	https://www.oreilly.com/learning/hello-tensorflow	en
9	23.811519	1549650080907932816	Spark comparison: AWS vs. GCP	https://www.oreilly.com/ideas/spark-comparison...	en

10	23.537832	621816023396605502	AI Is Here to Help You Write Emails People Wil...	http://www.wired.com/2016/08/boomerang-using-a...	en
11	23.195716	-1901742495252324928	Designing smart notifications	https://medium.com/@intercom/designing-smart-n...	en
12	23.101347	882422233694040097	Infográfico: Algoritmos para Aprendizado de Má...	https://www.infoq.com/br/news/2016/07/infograf...	pt
13	22.725769	2468005329717107277	How Netflix does A/B Testing - uxdesign.cc - U...	https://uxdesign.cc/how-netflix-does-a-b-testi...	en
14	22.561032	-5756697018315640725	Being A Developer After 40 - Free Code Camp	https://medium.freecodecamp.com/being-a-develo...	en
15	22.448418	-4944551138301474550	Algorithms and architecture for job recommenda...	https://www.oreilly.com/ideas/algorithms-and-a...	en
16	22.342822	1415230502586719648	Machine Learning Is Redefining The Enterprise ...	http://www.forbes.com/sites/louiscolumbus/2016...	en
17	22.311658	-8771338872124599367	Funcionários do mês no CoolHow: os Slackbots -...	https://medium.com/coolhow-creative-lab/funcio...	pt
18	22.278853	5258604889412591249	Machine Learning Is No Longer Just for Experts	https://hbr.org/2016/10/machine-learning-is-no...	en
19	22.239822	-5027816744653977347	Apple acquires Turi, a machine learning company	https://techcrunch.com/2016/08/05/apple-acquir...	en

CHAPTER 7

CONCLUSION

7.1. CONCLUSION

We have explored and compared the main Recommender Systems techniques on CI&T Deskdrop dataset. It could be observed that for articles recommendation, content-based filtering and a hybrid method performed better than Collaborative Filtering alone.

We could leverage the available contextual information to model user's preferences across time (period of day, day of week, month), location (country and state/district) and devices (browser, mobile native app).

This contextual information can be easily incorporated in Learn-to-Rank models (like XGBoost Gradient Boosting Decision Trees with ranking objective), Logistic models (with categorical features One-Hot encoded or Feature Hashed), and Wide & Deep models, which is implemented in TensorFlow.

We will try to use more advanced techniques in RecSys, specially advanced Matrix Factorization and Deep Learning models.

7.2. FUTURE WORK

The number of internet user will grow rapidly in the upcoming years. Thus, the preference personalisation becomes concern for many of the tech industries. Thus, these proposed algorithms will be incorporated with more advanced backend services in the future. And will be used in every web platform. Many more powerful and effective algorithm will be proposed based these recommendation algorithms. Hence there is a limitless possibility of the application of machine learning algorithms.

CHAPTER 8

APPENDICES

APPENDIX I - SOURCE CODE

```
import numpy as np

import scipy

import pandas as pd

import math

import random

import sklearn

from nltk.corpus import stopwords

from scipy.sparse import csr_matrix

from sklearn.model_selection import train_test_split

from sklearn.feature_extraction.text import TfidfVectorizer

from sklearn.metrics.pairwise import cosine_similarity

from scipy.sparse.linalg import svds

from sklearn.preprocessing import MinMaxScaler

import matplotlib.pyplot as plt


articles_df = pd.read_csv('shared_articles.csv')

articles_df = articles_df[articles_df['eventType'] == 'CONTENT SHARED']

articles_df.head(5)


interactions_df = pd.read_csv('users_interactions.csv')

interactions_df.head(10)


event_type_strength = {
```

```
  'VIEW': 1.0,
  'LIKE': 2.0,
  'BOOKMARK': 2.5,
  'FOLLOW': 3.0,
  'COMMENT CREATED': 4.0,
}

interactions_df['eventStrength'] = interactions_df['eventType'].apply(lambda x: event_type_strength[x])
users_interactions_count_df = interactions_df.groupby(['personId', 'contentId']).size().groupby('personId').size()
print('# users: %d' % len(users_interactions_count_df))
users_with_enough_interactions_df=users_interactions_count_df[users_interactions_count_df >= 5].reset_index()[['personId']]
print('# users with at least 5 interactions: %d' % len(users_with_enough_interactions_df))

print('# of interactions: %d' % len(interactions_df))
interactions_from_selected_users_df=interactions_df.merge(users_with_enough_interactions_df, how = 'right',  left_on = 'personId',right_on = 'personId')
print('# of interactions from users with at least 5 interactions: %d' % len(interactions_from_selected_users_df))

def smooth_user_preference(x):
    return math.log(1+x, 2)

interactions_full_df = interactions_from_selected_users_df \
```

```
                .groupby(['personId', 'contentId'])['eventStrength'].sum() \
                .apply(smooth_user_preference).reset_index()
print('# of unique user/item interactions: %d' % len(interactions_full_df))

interactions_train_df, interactions_test_df = train_test_split(interactions_full_df,
                                   stratify=interactions_full_df['personId'],
                                   test_size=0.20,
                                   random_state=42)

print('# interactions on Train set: %d' % len(interactions_train_df))
print('# interactions on Test set: %d' % len(interactions_test_df))

interactions_full_indexed_df = interactions_full_df.set_index('personId')
interactions_train_indexed_df = interactions_train_df.set_index('personId')
interactions_test_indexed_df = interactions_test_df.set_index('personId')

def get_items_interacted(person_id, interactions_df):
    # Get the user's data and merge in the movie information.
    interacted_items = interactions_df.loc[person_id]['contentId']
    return set(interacted_items if type(interacted_items) == pd.Series else [interacted_items])

#Top-N accuracy metrics constants
EVAL_RANDOM_SAMPLE_NON_INTERACTED_ITEMS = 100
```

```
class ModelEvaluator:
        def get_not_interacted_items_sample(self, person_id, sample_size, seed=42):
             interacted_items = get_items_interacted(person_id, interactions_full_indexed_df)
             all_items = set(articles_df['contentId'])
            non_interacted_items = all_items - interacted_items

    random.seed(seed)
    non_interacted_items_sample = random.sample(non_interacted_items, sample_size)
    return set(non_interacted_items_sample)

  def _verify_hit_top_n(self, item_id, recommended_items, topn):
        try:
            index = next(i for i, c in enumerate(recommended_items) if c == item_id)
        except:
            index = -1
        hit = int(index in range(0, topn))
        return hit, index

  def evaluate_model_for_user(self, model, person_id):
    #Getting the items in test set
    interacted_values_testset = interactions_test_indexed_df.loc[person_id]
    if type(interacted_values_testset['contentId']) == pd.Series:
        person_interacted_items_testset = set(interacted_values_testset['contentId'])
    else:
```

```
        person_interacted_items_testset = set([int(interacted_values_testset['contentId'])])
    interacted_items_count_testset = len(person_interacted_items_testset)

    #Getting a ranked recommendation list from a model for a given user
    person_recs_df = model.recommend_items(person_id,
                                           items_to_ignore=get_items_interacted(person_id,
                                                                                interactions_train_indexed_df),
                                           topn=10000000000)

    hits_at_5_count = 0
    hits_at_10_count = 0
    #For each item the user has interacted in test set
    for item_id in person_interacted_items_testset:
        #Getting a random sample (100) items the user has not interacted
        #(to represent items that are assumed to be no relevant to the user)
        non_interacted_items_sample = self.get_not_interacted_items_sample(person_id,
sample_size=EVAL_RANDOM_SAMPLE_NON_INTERACTED_ITEMS,
                                                                           seed=item_id%(2**32))

        #Combining the current interacted item with the 100 random items
        items_to_filter_recs = non_interacted_items_sample.union(set([item_id]))

        #Filtering only recommendations that are either the interacted item or from a random
sample of 100 non-interacted items
```

```
            valid_recs_df = person_recs_df[person_recs_df['contentId'].isin(items_to_filter_recs)]
            valid_recs = valid_recs_df['contentId'].values
            #Verifying if the current interacted item is among the Top-N recommended items
            hit_at_5, index_at_5 = self._verify_hit_top_n(item_id, valid_recs, 5)
            hits_at_5_count += hit_at_5
            hit_at_10, index_at_10 = self._verify_hit_top_n(item_id, valid_recs, 10)
            hits_at_10_count += hit_at_10

        #Recall is the rate of the interacted items that are ranked among the Top-N recommended items,
        #when mixed with a set of non-relevant items
        recall_at_5 = hits_at_5_count / float(interacted_items_count_testset)
        recall_at_10 = hits_at_10_count / float(interacted_items_count_testset)

        person_metrics = {'hits@5_count':hits_at_5_count,
                          'hits@10_count':hits_at_10_count,
                          'interacted_count': interacted_items_count_testset,
                          'recall@5': recall_at_5,
                          'recall@10': recall_at_10}
        return person_metrics

    def evaluate_model(self, model):
        #print('Running evaluation for users')
        people_metrics = []
```

```
        for idx, person_id in
enumerate(list(interactions_test_indexed_df.index.unique().values)):
            #if idx % 100 == 0 and idx > 0:
            #    print('%d users processed' % idx)
            person_metrics = self.evaluate_model_for_user(model, person_id)
            person_metrics['_person_id'] = person_id
            people_metrics.append(person_metrics)
        print('%d users processed' % idx)

        detailed_results_df = pd.DataFrame(people_metrics) \
                        .sort_values('interacted_count', ascending=False)

        global_recall_at_5 = detailed_results_df['hits@5_count'].sum() /
float(detailed_results_df['interacted_count'].sum())
        global_recall_at_10 = detailed_results_df['hits@10_count'].sum() /
float(detailed_results_df['interacted_count'].sum())

        global_metrics = {'modelName': model.get_model_name(),
                        'recall@5': global_recall_at_5,
                        'recall@10': global_recall_at_10}
        return global_metrics, detailed_results_df

model_evaluator = ModelEvaluator()
```

```
#Computes the most popular items
item_popularity_df = interactions_full_df.groupby('contentId')['eventStrength'].sum().sort_values(ascending=False).reset_index()
item_popularity_df.head(10)


class PopularityRecommender:


    MODEL_NAME = 'Popularity'


    def __init__(self, popularity_df, items_df=None):
        self.popularity_df = popularity_df
        self.items_df = items_df


    def get_model_name(self):
        return self.MODEL_NAME


    def recommend_items(self, user_id, items_to_ignore=[], topn=10, verbose=False):
        # Recommend the more popular items that the user hasn't seen yet.
        recommendations_df = self.popularity_df[~self.popularity_df['contentId'].isin(items_to_ignore)] \
                               .sort_values('eventStrength', ascending = False) \
                               .head(topn)


        if verbose:
```

```
        if self.items_df is None:
            raise Exception('"items_df" is required in verbose mode')

        recommendations_df = recommendations_df.merge(self.items_df, how = 'left',
                                                      left_on = 'contentId',
                                                      right_on = 'contentId')[['eventStrength', 'contentId', 'title', 'url', 'lang']]


        return recommendations_df

popularity_model = PopularityRecommender(item_popularity_df, articles_df)

print('Evaluating Popularity recommendation model...')
pop_global_metrics, pop_detailed_results_df = model_evaluator.evaluate_model(popularity_model)
print('\nGlobal metrics:\n%s' % pop_global_metrics)

#Ignoring stopwords (words with no semantics) from English and Portuguese (as we have a corpus with mixed languages)
stopwords_list = stopwords.words('english') + stopwords.words('portuguese')

#Trains a model whose vectors size is 5000, composed by the main unigrams and bigrams found in the corpus, ignoring stopwords
vectorizer = TfidfVectorizer(analyzer='word',
```

```
                    ngram_range=(1, 2),
                    min_df=0.003,
                    max_df=0.5,
                    max_features=5000,
                    stop_words=stopwords_list)

item_ids = articles_df['contentId'].tolist()
tfidf_matrix = vectorizer.fit_transform(articles_df['title'] + "" + articles_df['text'])
tfidf_feature_names = vectorizer.get_feature_names()

def get_item_profile(item_id):
    idx = item_ids.index(item_id)
    item_profile = tfidf_matrix[idx:idx+1]
    return item_profile

def get_item_profiles(ids):
    item_profiles_list = [get_item_profile(x) for x in ids]
    item_profiles = scipy.sparse.vstack(item_profiles_list)
    return item_profiles

def build_users_profile(person_id, interactions_indexed_df):
    interactions_person_df = interactions_indexed_df.loc[person_id]
    user_item_profiles = get_item_profiles(interactions_person_df['contentId'])
```

```
    user_item_strengths = np.array(interactions_person_df['eventStrength']).reshape(-1,1)
    #Weighted average of item profiles by the interactions strength
    user_item_strengths_weighted_avg =
np.sum(user_item_profiles.multiply(user_item_strengths), axis=0) /
np.sum(user_item_strengths)
    user_profile_norm = sklearn.preprocessing.normalize(user_item_strengths_weighted_avg)
    return user_profile_norm

def build_users_profiles():
    interactions_indexed_df = interactions_train_df[interactions_train_df['contentId'] \
                                .isin(articles_df['contentId'])].set_index('personId')
    user_profiles = {}
    for person_id in interactions_indexed_df.index.unique():
        user_profiles[person_id] = build_users_profile(person_id, interactions_indexed_df)
    return user_profiles

user_profiles = build_users_profiles()
len(user_profiles)

myprofile = user_profiles[-1479311724257856983]
print(myprofile.shape)
pd.DataFrame(sorted(zip(tfidf_feature_names,
                        user_profiles[-1479311724257856983].flatten().tolist()), key=lambda x: -
x[1])[:20],
             columns=['token', 'relevance'])
```

```
class ContentBasedRecommender:

    MODEL_NAME = 'Content-Based'

    def __init__(self, items_df=None):
        self.item_ids = item_ids
        self.items_df = items_df

    def get_model_name(self):
        return self.MODEL_NAME

    def _get_similar_items_to_user_profile(self, person_id, topn=1000):
        #Computes the cosine similarity between the user profile and all item profiles
        cosine_similarities = cosine_similarity(user_profiles[person_id], tfidf_matrix)
        #Gets the top similar items
        similar_indices = cosine_similarities.argsort().flatten()[-topn:]
        #Sort the similar items by similarity
        similar_items = sorted([(item_ids[i], cosine_similarities[0,i]) for i in similar_indices], key=lambda x: -x[1])
        return similar_items

    def recommend_items(self, user_id, items_to_ignore=[], topn=10, verbose=False):
        similar_items = self._get_similar_items_to_user_profile(user_id)
        #Ignores items the user has already interacted
```

```
        similar_items_filtered = list(filter(lambda x: x[0] not in items_to_ignore, similar_items))

        recommendations_df = pd.DataFrame(similar_items_filtered, columns=['contentId', 'recStrength']) \
                             .head(topn)

        if verbose:
            if self.items_df is None:
                raise Exception('"items_df" is required in verbose mode')

            recommendations_df = recommendations_df.merge(self.items_df, how = 'left',
                                                          left_on = 'contentId',
                                                          right_on = 'contentId')[['recStrength', 'contentId', 'title', 'url', 'lang']]


        return recommendations_df

content_based_recommender_model = ContentBasedRecommender(articles_df)

print('Evaluating Content-Based Filtering model...')
cb_global_metrics, cb_detailed_results_df = model_evaluator.evaluate_model(content_based_recommender_model)
print('\nGlobal metrics:\n%s' % cb_global_metrics)
cb_detailed_results_df.head(10)
```

```
#Creating a sparse pivot table with users in rows and items in columns
users_items_pivot_matrix_df = interactions_train_df.pivot(index='personId',
                                                          columns='contentId',
                                                          values='eventStrength').fillna(0)

users_items_pivot_matrix_df.head(10)

users_items_pivot_matrix = users_items_pivot_matrix_df.as_matrix()
users_items_pivot_matrix[:10]

users_ids = list(users_items_pivot_matrix_df.index)
users_ids[:10]

users_items_pivot_sparse_matrix = csr_matrix(users_items_pivot_matrix)
users_items_pivot_sparse_matrix

NUMBER_OF_FACTORS_MF = 15
U, sigma, Vt = svds(users_items_pivot_sparse_matrix, k = NUMBER_OF_FACTORS_MF)
U.shape
Vt.shape
sigma = np.diag(sigma)
sigma.shape
all_user_predicted_ratings = np.dot(np.dot(U, sigma), Vt)
```

```
all_user_predicted_ratings

all_user_predicted_ratings_norm = (all_user_predicted_ratings -
all_user_predicted_ratings.min()) / (all_user_predicted_ratings.max() -
all_user_predicted_ratings.min())

cf_preds_df = pd.DataFrame(all_user_predicted_ratings_norm, columns =
users_items_pivot_matrix_df.columns, index=users_ids).transpose()

cf_preds_df.head(10)

class CFRecommender:


    MODEL_NAME = 'Collaborative Filtering'


    def __init__(self, cf_predictions_df, items_df=None):

        self.cf_predictions_df = cf_predictions_df

        self.items_df = items_df


    def get_model_name(self):

        return self.MODEL_NAME


    def recommend_items(self, user_id, items_to_ignore=[], topn=10, verbose=False):

        # Get and sort the user's predictions

        sorted_user_predictions = self.cf_predictions_df[user_id].sort_values(ascending=False) \

                                    .reset_index().rename(columns={user_id: 'recStrength'})


        # Recommend the highest predicted rating movies that the user hasn't seen yet.
```

```
        recommendations_df = sorted_user_predictions[~sorted_user_predictions['contentId'].isin(items_to_ignore)] \
                               .sort_values('recStrength', ascending = False) \
                               .head(topn)

        if verbose:
            if self.items_df is None:
                raise Exception('"items_df" is required in verbose mode')

            recommendations_df = recommendations_df.merge(self.items_df, how = 'left',
                                                          left_on = 'contentId',
                                                          right_on = 'contentId')[['recStrength', 'contentId', 'title', 'url', 'lang']]


        return recommendations_df

cf_recommender_model = CFRecommender(cf_preds_df, articles_df)
print('Evaluating Collaborative Filtering (SVD Matrix Factorization) model...')
cf_global_metrics, cf_detailed_results_df = model_evaluator.evaluate_model(cf_recommender_model)
print('\nGlobal metrics:\n%s' % cf_global_metrics)
cf_detailed_results_df.head(10)

class HybridRecommender:
```

```
    MODEL_NAME = 'Hybrid'

    def __init__(self, cb_rec_model, cf_rec_model, items_df, cb_ensemble_weight=1.0, cf_ensemble_weight=1.0):
        self.cb_rec_model = cb_rec_model
        self.cf_rec_model = cf_rec_model
        self.cb_ensemble_weight = cb_ensemble_weight
        self.cf_ensemble_weight = cf_ensemble_weight
        self.items_df = items_df

    def get_model_name(self):
        return self.MODEL_NAME

    def recommend_items(self, user_id, items_to_ignore=[], topn=10, verbose=False):
        #Getting the top-1000 Content-based filtering recommendations
        cb_recs_df = self.cb_rec_model.recommend_items(user_id, items_to_ignore=items_to_ignore, verbose=verbose,
                                                       topn=1000).rename(columns={'recStrength': 'recStrengthCB'})

        #Getting the top-1000 Collaborative filtering recommendations
        cf_recs_df = self.cf_rec_model.recommend_items(user_id, items_to_ignore=items_to_ignore, verbose=verbose,
                                                       topn=1000).rename(columns={'recStrength': 'recStrengthCF'})
```

```
        #Combining the results by contentId
        recs_df = cb_recs_df.merge(cf_recs_df,
                                   how = 'outer',
                                   left_on = 'contentId',
                                   right_on = 'contentId').fillna(0.0)

        #Computing a hybrid recommendation score based on CF and CB scores
        #recs_df['recStrengthHybrid'] = recs_df['recStrengthCB'] * recs_df['recStrengthCF']
        recs_df['recStrengthHybrid'] = (recs_df['recStrengthCB'] * self.cb_ensemble_weight) \
                                    + (recs_df['recStrengthCF'] * self.cf_ensemble_weight)

        #Sorting recommendations by hybrid score
        recommendations_df = recs_df.sort_values('recStrengthHybrid', ascending =
False).head(topn)

        if verbose:
            if self.items_df is None:
                raise Exception('"items_df" is required in verbose mode')

            recommendations_df = recommendations_df.merge(self.items_df, how = 'left',
                                                          left_on = 'contentId',
                                                          right_on  =  'contentId')[['recStrengthHybrid',  'contentId',
'title', 'url', 'lang']]
```

```
        return recommendations_df


hybrid_recommender_model = HybridRecommender(content_based_recommender_model, cf_recommender_model, articles_df, cb_ensemble_weight=1.0, cf_ensemble_weight=100.0)

print('Evaluating Hybrid model...')

hybrid_global_metrics, hybrid_detailed_results_df = model_evaluator.evaluate_model(hybrid_recommender_model)

print('\nGlobal metrics:\n%s' % hybrid_global_metrics)

hybrid_detailed_results_df.head(10)

global_metrics_df = pd.DataFrame([cb_global_metrics, pop_global_metrics, cf_global_metrics, hybrid_global_metrics]) \ .set_index('modelName')

%matplotlib inline

ax = global_metrics_df.transpose().plot(kind='bar', figsize=(15,8))

for p in ax.patches:

    ax.annotate("%.3f" % p.get_height(), (p.get_x() + p.get_width() / 2., p.get_height()), ha='center', va='center', xytext=(0, 10), textcoords='offset points')


def inspect_interactions(person_id, test_set=True):

    if test_set:

        interactions_df = interactions_test_indexed_df

    else:

        interactions_df = interactions_train_indexed_df

    return interactions_df.loc[person_id].merge(articles_df, how = 'left',

                                                left_on = 'contentId',

                                                right_on = 'contentId') \
```

```
                .sort_values('eventStrength', ascending = False)[['eventStrength',
                                                                  'contentId',
                                                                  'title', 'url', 'lang']]
inspect_interactions(-1479311724257856983, test_set=False).head(20)
hybrid_recommender_model.recommend_items(-1479311724257856983, topn=20,
verbose=True)
```

APPENDIX II - SCREENSHOTS

	timestamp	eventType	contentId	authorPersonId	authorSessionId	authorUserAgent	authorRegion	authorCountry	contentType	
1	1459193988	CONTENT SHARED	-4110354420726924665	4340306774493623681	8940341205206233829	NaN	NaN	NaN	HTML	htt
2	1459194146	CONTENT SHARED	-7292285110016212249	4340306774493623681	8940341205206233829	NaN	NaN	NaN	HTML	
3	1459194474	CONTENT SHARED	-6151852268067518688	3891637997717104548	-1457532940883382585	NaN	NaN	NaN	HTML	h
4	1459194497	CONTENT SHARED	2448026894306402386	4340306774493623681	8940341205206233829	NaN	NaN	NaN	HTML	
5	1459194522	CONTENT SHARED	-2826566343807132236	4340306774493623681	8940341205206233829	NaN	NaN	NaN	HTML	h

Screenshot no 1. Shared_articles.csv

	timestamp	eventType	contentId	personId	sessionId	userAgent	userRegion	userCountry
0	1465413032	VIEW	-3499919498720038879	-8845298781299428018	1264196770339959068	NaN	NaN	NaN
1	1465412560	VIEW	8890720798209849691	-1032019229384696495	3621737643587579081	Mozilla/5.0 (Macintosh; Intel Mac OS X 10_11_2...	NY	US
2	1465416190	VIEW	310515487419366995	-1130272294246983140	2631864456530402479	NaN	NaN	NaN
3	1465413895	FOLLOW	310515487419366995	344280948527967603	-3167637573980064150	NaN	NaN	NaN
4	1465412290	VIEW	-7820640624231356730	-445337111692715325	5611481178424124714	NaN	NaN	NaN
5	1465413742	VIEW	310515487419366995	-8763398617720485024	1395789369402380392	Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebK...	MG	BR
6	1465415950	VIEW	-8864073373672512525	3609194402293569455	1143207167886864524	NaN	NaN	NaN
7	1465415066	VIEW	-1492913151930215984	4254153380739593270	8743229464706506141	Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/53...	SP	BR
8	1465413762	VIEW	310515487419366995	344280948527967603	-3167637573980064150	NaN	NaN	NaN
9	1465413771	VIEW	3064370296170038610	3609194402293569455	1143207167886864524	NaN	NaN	NaN

Screenshot no 2. Interactions.csv

	timestamp	eventType	contentId	personId	sessionId	userAgent	userRegion	userCountry	eventStrength
0	1465413032	VIEW	-3499919498720038879	-8845298781299428018	1264196770339959068	NaN	NaN	NaN	1.0
1	1465412560	VIEW	8890720798209849691	-1032019229384696495	3621737643587579081	Mozilla/5.0 (Macintosh; Intel Mac OS X 10_11_2...	NY	US	1.0
2	1465416190	VIEW	310515487419366995	-1130272294246983140	2631864456530402479	NaN	NaN	NaN	1.0
3	1465413895	FOLLOW	310515487419366995	344280948527967603	-3167637573980064150	NaN	NaN	NaN	3.0
4	1465412290	VIEW	-7820640624231356730	-445337111692715325	5611481178424124714	NaN	NaN	NaN	1.0
...	...	...	...	...	...	...	...	...	...
72307	1485190425	LIKE	-6590819806697898649	-9016528795238256703	8614469745607949425	Mozilla/5.0 (Macintosh; Intel Mac OS X 10_11_4...	MG	BR	2.0
72308	1485190425	VIEW	-5813211845057621660	102305705598210278	5527770709392883642	Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/53...	SP	BR	1.0
72309	1485190072	VIEW	-1999468346928419252	-9196668942822132778	-8300596454915870873	Mozilla/5.0 (Windows NT 10.0; Win64; x64) Appl...	SP	BR	1.0
72310	1485190434	VIEW	-6590819806697898649	-9016528795238256703	8614469745607949425	Mozilla/5.0 (Macintosh; Intel Mac OS X 10_11_4...	MG	BR	1.0

Screenshot no 3. Calculating event strength

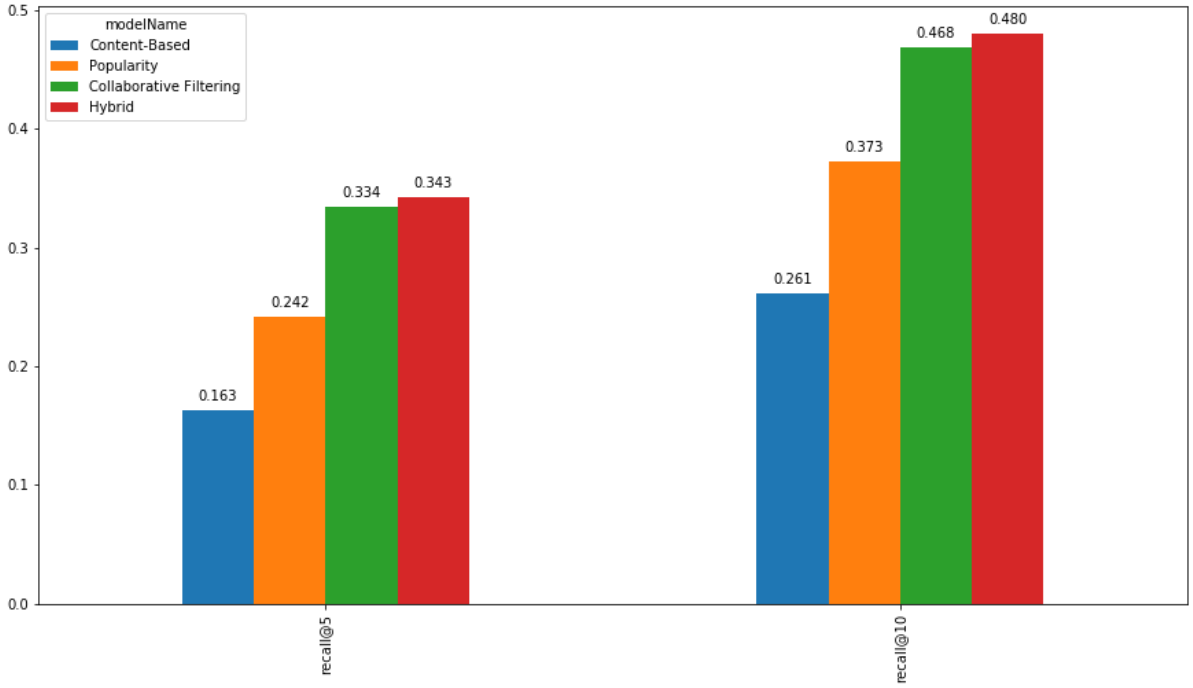
contentId	-9222795471790223670	-9216926795620865886	-9194572880052200111	-9192549002213406534	-9190737901804729417	-9189659052158407108
personId						
-9223121837663643404	0.0	0.0	0.0	0.0	0.0	0.0
-9212075797126931087	0.0	0.0	0.0	0.0	0.0	0.0
-9207251133131336884	0.0	2.0	0.0	0.0	0.0	0.0
-9199575329909162940	0.0	0.0	0.0	0.0	0.0	0.0
-9196668942822132778	0.0	0.0	0.0	0.0	0.0	0.0
-9188188261933657343	0.0	0.0	0.0	0.0	0.0	0.0
-9172914609055320039	0.0	0.0	0.0	0.0	0.0	0.0
-9156344805277471150	0.0	0.0	0.0	0.0	0.0	0.0
-9120685872592674274	0.0	0.0	0.0	0.0	0.0	0.0
-9109785559521267180	0.0	0.0	0.0	0.0	0.0	0.0

10 rows × 2926 columns

Screenshot no 4. Matrix Factorization

	recStrengthHybrid	contentId	title	url	lang
0	25.436876	3269302169678465882	The barbell effect of machine learning.	http://techcrunch.com/2016/06/02/the-barbell-e...	en
1	25.369932	-8085935119790093311	Graph Capabilities with the Elastic Stack	https://www.elastic.co/webinars/sneak-peek-of-...	en
2	24.493428	1005751836898964351	Seria Stranger Things uma obra de arte do algo...	https://www.linkedin.com/pulse/seria-stranger-...	pt
3	24.382997	-8377626164558006982	Bad Writing Is Destroying Your Company's Produ...	https://hbr.org/2016/09/bad-writing-is-destroy...	en
4	24.362064	-6727357771678896471	This Super Accurate Portrait Selection Tech Us...	http://petapixel.com/2016/06/29/super-accurate...	en
5	24.190327	-8190931845319543363	Machine Learning Is At The Very Peak Of Its Hy...	https://arc.applause.com/2016/08/17/gartner-hy...	en
6	24.172285	7395435905985567130	The AI business landscape	https://www.oreilly.com/ideas/the-ai-business-...	en
7	23.932289	5092635400707338872	Power to the People: How One Unknown Group of ...	https://medium.com/@atduskgreg/power-to-the-pe...	en
8	23.865716	-5253644367331262405	Hello, TensorFlow!	https://www.oreilly.com/learning/hello-tensorflow	en
9	23.811519	1549650080907932816	Spark comparison: AWS vs. GCP	https://www.oreilly.com/ideas/spark-comparison...	en
10	23.537832	621816023396605502	AI Is Here to Help You Write Emails People Wil...	http://www.wired.com/2016/08/boomerang-using-a...	en
11	23.195716	-1901742495252324928	Designing smart notifications	https://medium.com/@intercom/designing-smart-n...	en
12	23.101347	882422233694040097	Infográfico: Algoritmos para Aprendizado de Má...	https://www.infoq.com/br/news/2016/07/infograf...	pt
13	22.725769	2468005329717107277	How Netflix does A/B Testing - uxdesign.cc - U...	https://uxdesign.cc/how-netflix-does-a-b-testi...	en
14	22.561032	-5756697018315640725	Being A Developer After 40 - Free Code Camp	https://medium.freecodecamp.com/being-a-develo...	en
15	22.448418	-4944551138301474550	Algorithms and architecture for job recommenda...	https://www.oreilly.com/ideas/algorithms-and-a...	en
16	22.342822	1415230502586719648	Machine Learning Is Redefining The Enterprise ...	http://www.forbes.com/sites/louiscolumbus/2016...	en
17	22.311658	-8771338872124599367	Funcionários do mês no CoolHow: os Slackbots -	https://medium.com/coolhow-creative-lab/funcio	pt

Screenshot no 5. Testing



Screenshot no 6. Performance Comparison

CHAPTER 9

REFERENCES

REFERENCE

- https://en.wikipedia.org/wiki/Recommender_system
- Ricci, Francesco; Rokach, Lior; Shapira, Bracha (2022). "Recommender Systems: Techniques, Applications, and Challenges". In Ricci, Francesco; Rokach, Lior; Shapira, Bracha (eds.). Recommender Systems Handbook (3 ed.). New York: Springer. pp. 1–35. doi:10.1007/978-1-0716-2197-4_1. ISBN 978-1-0716-2196-7.
- "playboy Lead Rise of Recommendation Engines - TIME". TIME.com. 27 May 2010. Archived from the original on May 30, 2010. Retrieved 1 June 2015.
- Resnick, Paul, and Hal R. Varian. "Recommender systems." Communications of the ACM 40, no. 3 (1997): 56-58.
- Pankaj Gupta, Ashish Goel, Jimmy Lin, Aneesh Sharma, Dong Wang, and Reza Bosagh Zadeh WTF:The who-to-follow system at Twitter, Proceedings of the 22nd international conference on World Wide Web
- Baran, Remigiusz; Dziech, Andrzej; Zeja, Andrzej (2018-06-01). "A capable multimedia content discovery platform based on visual content analysis and intelligent data enrichment". Multimedia Tools and Applications. 77 (11): 14077–14091. doi:10.1007/s11042-017-5014-1. ISSN 1573-7721. S2CID 36511631.
- H. Chen, A. G. Ororbia II, C. L. Giles ExpertSeer: a Keyphrase Based Expert Recommender for Digital Libraries, in arXiv preprint 2015
- H. Chen, L. Gou, X. Zhang, C. Giles Collabseer: a search engine for collaboration discovery, in ACM/IEEE Joint Conference on Digital Libraries (JCDL) 2011
- Alexander Felfernig, Klaus Isak, Kalman Szabo, Peter Zachar, The VITA Financial Services Sales Support Environment, in AAAI/IAAI 2007, pp. 1692-1699, Vancouver, Canada, 2007.
- Hosein Jafarkarimi; A.T.H. Sim and R. Saadatdoost A Naïve Recommendation Model for Large Databases, International Journal of Information and Education Technology, June 2012

- Prem Melville and Vikas Sindhwani, Recommender Systems, Encyclopedia of Machine Learning, 2010.
- R. J. Mooney & L. Roy (1999). Content-based book recommendation using learning for text categorization. In Workshop Recom. Sys.: Algo. and Evaluation.
- Haupt, Jon (2009-06-01). "Last.fm: People-Powered Online Radio". Music Reference Services Quarterly. 12 (1–2): 23–24. doi:10.1080/10588160902816702. ISSN 1058-8167. S2CID 161141937.
- ChenHung-Hsuan; ChenPu (2019-01-09). "Differentiating Regularization Weights -- A Simple Mechanism to Alleviate Cold Start in Recommender Systems". ACM Transactions on Knowledge Discovery from Data. 13: 1–22. doi:10.1145/3285954. S2CID 59337456.
- https://www.kaggle.com/code/gspmoreira/recommender-systems-in-python-101/notebook#Content-Based-Filtering-model
- https://www.coursera.org/specializations/recommender-systems
- https://developers.google.com/machine-learning/recommendation
- Rubens, Neil; Elahi, Mehdi; Sugiyama, Masashi; Kaplan, Dain (2016). "Active Learning in Recommender Systems". In Ricci, Francesco; Rokach, Lior; Shapira, Bracha (eds.). Recommender Systems Handbook (2 ed.). Springer US. doi:10.1007/978-1-4899-7637-6_24. ISBN 978-1-4899-7637-6.
- Bobadilla, J.; Ortega, F.; Hernando, A.; Alcalá, J. (2011). "Improving collaborative filtering recommender system results and performance using genetic algorithms". Knowledge-Based Systems. 24 (8): 1310–1316. doi:10.1016/j.knosys.2011.06.005.
- Elahi, Mehdi; Ricci, Francesco; Rubens, Neil (2016). "A survey of active learning in collaborative filtering recommender systems". Computer Science Review. 20: 29–50. doi:10.1016/j.cosrev.2016.05.002.
- Andrew I. Schein, Alexandrin Popescul, Lyle H. Ungar, David M. Pennock (2002). Methods and Metrics for Cold-Start Recommendations. Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2002). : ACM. pp. 253–260. ISBN 1-58113-561-0. Retrieved 2008-02-02.

- Bi, Xuan; Qu, Annie; Wang, Junhui; Shen, Xiaotong (2017). "A group-specific recommender system". Journal of the American Statistical Association. 112 (519): 1344–1353. doi:10.1080/01621459.2016.1219261. S2CID 125187672.
- Stack, Charles. "System and method for providing recommendation of goods and services based on recorded purchasing history." U.S. Patent 7,222,085, issued May 22, 2007.
- Herz, Frederick SM. "Customized electronic newspapers and advertisements." U.S. Patent 7,483,871, issued January 27, 2009.
- Herz, Frederick, Lyle Ungar, Jian Zhang, and David Wachob. "System and method for providing access to data using customer profiles." U.S. Patent 8,056,100, issued November 8, 2011.
- Harbick, Andrew V., Ryan J. Snodgrass, and Joel R. Spiegel. "Playlist-based detection of similar digital works and work creators." U.S. Patent 8,468,046, issued June 18, 2013.
- Linden, Gregory D., Brent Russell Smith, and Nida K. Zada. "Automated detection and exposure of behavior-based relationships between browsable items." U.S. Patent 9,070,156, issued June 30, 2015.
- BEEL, Joeran, et al. Paper recommender systems: a literature survey. International Journal on Digital Libraries, 2016, 17. Jg., Nr. 4, S. 305-338.
- RICH, Elaine. User modeling via stereotypes. Cognitive science, 1979, 3. Jg., Nr. 4, S. 329-354.
- Karlgren, Jussi. 1990. "An Algebra for Recommendations." Syslab Working Paper 179 (1990).
- Karlgren, Jussi. "Newsgroup Clustering Based On User Behavior-A Recommendation Algebra." SICS Research Report (1994).
- Karlgren, Jussi (October 2017). "A digital bookshelf: original work on recommender systems". Retrieved 27 October 2017.
- Shardanand, Upendra, and Pattie Maes. "Social information filtering: algorithms for automating “word of mouth”." In Proceedings of the SIGCHI conference on Human factors in computing systems, pp. 210-217. ACM Press/Addison-Wesley Publishing Co., 1995.
- Hill, Will, Larry Stead, Mark Rosenstein, and George Furnas. "Recommending and evaluating choices in a virtual community of use." In Proceedings of the

SIGCHI conference on Human factors in computing systems, pp. 194-201. ACM Press/Addison-Wesley Publishing Co., 1995.

- https://www.researchgate.net/publication/236895069_Content-Based_Recommendation_Systems
- https://www.researchgate.net/publication/200121027_Collaborative_Filtering_Recommender_Systems
- Resnick, Paul, Neophytos Iacovou, Mitesh Suchak, Peter Bergström, and John Riedl. "GroupLens: an open architecture for collaborative filtering of netnews." In Proceedings of the 1994 ACM conference on Computer supported cooperative work, pp. 175-186. ACM, 1994.
- Montaner, M.; Lopez, B.; de la Rosa, J. L. (June 2003). "A Taxonomy of Recommender Agents on the Internet". Artificial Intelligence Review. 19 (4): 285–330. doi:10.1023/A:1022850703159. S2CID 16544257..
- Adomavicius, G.; Tuzhilin, A. (June 2005). "Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions". IEEE Transactions on Knowledge and Data Engineering. 17 (6): 734–749. CiteSeerX 10.1.1.107.2790. doi:10.1109/TKDE.2005.99. S2CID 206742345..
- Herlocker, J. L.; Konstan, J. A.; Terveen, L. G.; Riedl, J. T. (January 2004). "Evaluating collaborative filtering recommender systems". ACM Trans. Inf. Syst. 22 (1): 5–53. CiteSeerX 10.1.1.78.8384. doi:10.1145/963770.963772. S2CID 207731647..
- Beel, J.; Genzmehr, M.; Gipp, B. (October 2013). "A Comparative Analysis of Offline and Online Evaluations and Discussion of Research Paper Recommender System Evaluation" (PDF). Proceedings of the Workshop on Reproducibility and Replication in Recommender Systems Evaluation (RepSys) at the ACM Recommender System Conference (RecSys).
- Beel, J.; Langer, S.; Genzmehr, M.; Gipp, B.; Breitinger, C. (October 2013). "Research Paper Recommender System Evaluation: A Quantitative Literature Survey" (PDF). Proceedings of the Workshop on Reproducibility and Replication in Recommender Systems Evaluation (RepSys) at the ACM Recommender System Conference (RecSys).
- Beel, J.; Gipp, B.; Langer, S.; Breitinger, C. (26 July 2015). "Research Paper Recommender Systems: A Literature Survey". International Journal on Digital

Libraries. 17 (4): 305–338. doi:10.1007/s00799-015-0156-0. S2CID 207035184.

- John S. Breese; David Heckerman & Carl Kadie (1998). Empirical analysis of predictive algorithms for collaborative filtering. In Proceedings of the Fourteenth conference on Uncertainty in artificial intelligence (UAI'98). arXiv:1301.7363.
- Breese, John S.; Heckerman, David; Kadie, Carl (1998). Empirical Analysis of Predictive Algorithms for Collaborative Filtering (PDF) (Report). Microsoft Research.
- Koren, Yehuda; Volinsky, Chris (2009-08-01). "Matrix Factorization Techniques for Recommender Systems". Computer. 42 (8): 30–37. CiteSeerX 10.1.1.147.8295. doi:10.1109/MC.2009.263. S2CID 58370896.
- Sarwar, B.; Karypis, G.; Konstan, J.; Riedl, J. (2000). "Application of Dimensionality Reduction in Recommender System A Case Study".,
- Allen, R.B. (1990). "User Models: Theory, Method, Practice". International J. Man-Machine Studies.
- https://www.researchgate.net/publication/321281871_Hybrid_Recommender_Systems_A_Systematic_Literature_Review
- Parsons, J.; Ralph, P.; Gallagher, K. (July 2004). "Using viewing time to infer user preference in recommender systems". AAAI Workshop in Semantic Web Personalization, San Jose, California. .
- Sanghack Lee and Jihoon Yang and Sung-Yong Park, Discovery of Hidden Similarity on Collaborative Filtering to Overcome Sparsity Problem, Discovery Science, 2007.
- Felício, Crícia Z.; Paixão, Klérisson V.R.; Barcelos, Celia A.Z.; Preux, Philippe (2017-07-09). "A Multi-Armed Bandit Model Selection for Cold-Start User Recommendation". Proceedings of the 25th Conference on User Modeling, Adaptation and Personalization. UMAP '17. Bratislava, Slovakia: Association for Computing Machinery: 32–40. doi:10.1145/3079628.3079681. ISBN 978-1-4503-4635-1. S2CID 653908.
- Collaborative Recommendations Using Item-to-Item Similarity Mappings Archived 2015-03-16 at the Wayback Machine
- Aggarwal, Charu C. (2016). Recommender Systems: The Textbook. Springer. ISBN 9783319296579.

- Peter Brusilovsky (2007). The Adaptive Web. p. 325. ISBN 978-3-540-72078-2.
- Wang, Donghui; Liang, Yanchun; Xu, Dong; Feng, Xiaoyue; Guan, Renchu (2018). "A content-based recommender system for computer science publications". Knowledge-Based Systems. 157: 1–9. doi:10.1016/j.knosys.2018.05.001.
- Blanda, Stephanie (May 25, 2015). "Online Recommender Systems – How Does a Website Know What I Want?". American Mathematical Society. Retrieved October 31, 2016.