

EGE UNIVERSITY

COMPUTER ENGINEERING DEPARTMENT

**ARTIFICIAL INTELIGENCE & DEEP LEARNING**

**PROJECT 2**

Prepared By:

05180000017 – Deniz YÜREKDELER

Delivery Date

22.05.2020

Contents

[Table of Figures 3](#_Toc40798152)

[Introduction 4](#_Toc40798153)

[Problem and Methods 4](#_Toc40798154)

[Artificial Intelligence 5](#_Toc40798155)

[Recommender System 5](#_Toc40798156)

[Collaborative Filtering 5](#_Toc40798157)

[Nearest Neighbourhood 6](#_Toc40798158)

[Matrix Factorization 7](#_Toc40798159)

[Optimization 8](#_Toc40798160)

[Content Based Filtering 8](#_Toc40798161)

[Application Programming Interface 8](#_Toc40798162)

[WEB API 9](#_Toc40798163)

[Similar Works 9](#_Toc40798164)

[Designing Activity-aware Recommender Systems for Operating Rooms 9](#_Toc40798165)

[Design of Physical Activity Recommendation System 9](#_Toc40798166)

[Solution Development 10](#_Toc40798167)

[Experimental Work 22](#_Toc40798168)

[DATASET 22](#_Toc40798169)

[LIBRARIES 22](#_Toc40798170)

# Table of Figures

[Figure 1 Nearest Neighbourhood Similarity Formula 6](#_Toc40798127)

[Figure 2 Pearson Correlation & Cosine Similarity Formulas 7](#_Toc40798128)

[Figure 3 Singular Value Decomposition Visualization 7](#_Toc40798129)

[Figure 4 Importing Libraries and Creating Flask Application and Defining a Route 11](#_Toc40798130)

[Figure 5 Reading CSV Formatted Files and Loading Them as Pandas Dataframes 11](#_Toc40798131)

[Figure 6 Merging Dataframe on "activityId" and Calculating Mean Ratings & Rating Counts 12](#_Toc40798132)

[Figure 7 Result Dataframe after “ratings\_data” & “activity\_names” Dataframes are Merged on “activityId” 12](#_Toc40798133)

[Figure 8 Plotting Two Histograms in Seperate Matplotlib Figures 12](#_Toc40798134)

[Figure 9 Histogram that Shows Spread of Ratings 13](#_Toc40798135)

[Figure 10 Histogram that Shows Spread of Rating Counts 14](#_Toc40798136)

[Figure 11 Creating Rating Matrix and Calculating Pairwise Correlation 14](#_Toc40798137)

[Figure 12 Dropping Empty Cells on Ratings Matrix and Filtering Correlations with Minimum 2 Rating Counts 15](#_Toc40798138)

[Figure 13 Correlation Dataframe after It Undergoes Operations in Figure 12 15](#_Toc40798139)

[Figure 14 Python Method that Finds the Current Hour of Day and Season 16](#_Toc40798140)

[Figure 15 Python Method that Checks whether the Activity Fits in with Current Hour and Season 16](#_Toc40798141)

[Figure 16 Loop which Calls Supportive Methods for each Activity 17](#_Toc40798142)

[Figure 17 Python Code that has Changed to Implement Hybrid System 18](#_Toc40798143)

[Figure 18 Creating a New Dataframe for Content Based Filtering 18](#_Toc40798144)

[Figure 19 Extracting Keywords of each Activity in Dataframe 19](#_Toc40798145)

[Figure 20 Initializing Count Matrix to Calculate Cosine Similarity 19](#_Toc40798146)

[Figure 21 Sorting the Count Matrix and Selecting Top 5 Activities 20](#_Toc40798147)

[Figure 22 The End of the Collaborative Filtering Method 20](#_Toc40798148)

[Figure 23 Options of Right Cliking on Code Page 21](#_Toc40798149)

[Figure 24 Console Output of Running the Flask Application 21](#_Toc40798150)

[Figure 25 Visualising of Recommender System’s Result in Web Browser 21](#_Toc40798151)

# Introduction

With the growth of mobile phones and internet in recent years, the demand to mobile applications has greatly increased. Users expect much from the mobile applications and they want applications to be able to do what a human can do. Since the technology is evolving and time is getting more and more important aspect in our lives, people tend to handle many operations in one place in a fast way to save time and energy.

Therefore, it is no wonder that recommendation systems are getting extremely popular over the decade. Users need applications that can act and do things as if it is another human being. Therefore, this article aims to present a design for an activity/hobby recommender system that has been built as an application-programming-interface (API) so that mobile applications or web applications can use this recommender platform free and give recommendations to the users.

# Problem and Methods

For this article, a hybrid recommender system API will be built, This recommender API uses artificial intelligence in terms of collaborative filtering mixed with content based filtering to form the hybrid system. Recommender system will use machine learning’s sub-field natural language processing on content based filtering. This recommender system will be written in Python’s Flask web interface and once it is complete it will serve other applications/users as a web API. The API structure in this article is kept as simple as possible in purpose of not to detail the web programming but focus on the artificial intelligence instead.

## Artificial Intelligence

Artificial Intelligence (AI) is a demonstration of a machine’s intelligence which is completely opposite of natural intelligence that belongs to living beings. Artificial intelligence can be simplified as, a machine that is aware of its environment to take proper actions to reach its goal successfully. Artificial intelligence is a broad concept, though it is mostly used and referred to computers mimicking human learning/problem solving methods. Modern machine capabilities generally classified as AI include successfully [understanding human speech](http://wiki.pinsify.xyz/index.php?q=aHR0cHM6Ly9lbi53aWtpcGVkaWEub3JnL3dpa2kvTmF0dXJhbF9sYW5ndWFnZV91bmRlcnN0YW5kaW5n), competing at the highest level in [strategic game](http://wiki.pinsify.xyz/index.php?q=aHR0cHM6Ly9lbi53aWtpcGVkaWEub3JnL3dpa2kvU3RyYXRlZ2ljX2dhbWU) systems [autonomously operating cars](http://wiki.pinsify.xyz/index.php?q=aHR0cHM6Ly9lbi53aWtpcGVkaWEub3JnL3dpa2kvQXV0b25vbW91c19jYXI), intelligent routing in [content delivery networks](http://wiki.pinsify.xyz/index.php?q=aHR0cHM6Ly9lbi53aWtpcGVkaWEub3JnL3dpa2kvQ29udGVudF9kZWxpdmVyeV9uZXR3b3Jr), and [military simulations](http://wiki.pinsify.xyz/index.php?q=aHR0cHM6Ly9lbi53aWtpcGVkaWEub3JnL3dpa2kvTWlsaXRhcnlfc2ltdWxhdGlvbnM) (Artificial Intelligence, 2020).

Artificial intelligence has a long history, starting in 1955 as an academic discipline. Since the beginning, it has passed through optimism and winter phases and split into sub-fields based on technical considerations such as machine learning, artificial neural networks etc. Artificial intelligence has many real life examples, such as SIRI of Apple Inc. that is a text or audio based assistant that can handle some tasks for the user such as calling a person from contacts list. Another example is driverless vehicles, which uses image-processing techniques to follow the road lines or detect objects in any directions of the vehicle while going or parking.

## Recommender System

A recommender system is a subclass of information filtering system that seeks to predict the ”rating” or ”preference” a user would give to an item (Recommmender System, 2020). They are primarily used in commercial applications. Recommender systems most commonly used in playlist generators for video and music services like Netflix, YouTube and Spotify, product recommenders for services such as Amazon, or content recommenders for social media platforms such as Facebook and Twitter. These systems can operate using a single input, like music, or multiple inputs within and across platforms like news, books, and search queries. There are also popular recommender systems for specific topics like restaurants and online dating. Recommender systems have also been developed to explore research articles and experts, collaborators, and financial services.

Recommender systems generally use one of collaborative filtering or content based filtering but can also use these two or others altogether and create hybrid systems.

## Collaborative Filtering

Just as machine learning techniques, recommender systems predicts and recommends items based on user’s historical behaviour. Collaborative filtering can be generally divided in user-based and item-based filtering. The historical behaviour can be explicit rating or implicit rating. Explicit rating can be directly giving a point or rating to an item. Implicit rating on the other hand can be the items viewed, purchased. Collaborative filtering only needs user’s historical preference for items. The logic of this filtering says that if two users gave similar behaviour to some items they will give same behaviour on another set of items too.

Collaborative filtering uses different algorithms to make the prediction which is a complicated statistical calculation. There are Nearest Neighbourhood, Matrix Factorization and Optimization.

### Nearest Neighbourhood

Nearest neighbourhood algorithm is the standard method of collaborative filtering. For user based filtering, a target user will receive recommendations and some number of top similar other users. Nearest neighbourhood takes the weighted average of ratings from the similar users and subtracts the average ratings of each user to avoid bias (Introduction to Recommender System, 2020). Without knowing anything about items and users themselves, it is assumed that two users are similar when they give the same item similar ratings.

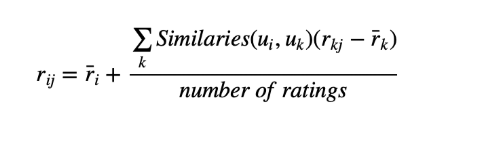


Figure Nearest Neighbourhood Similarity Formula

For calculating the similarity, there are two options one if Pearson correlation and other is cosine similarity.

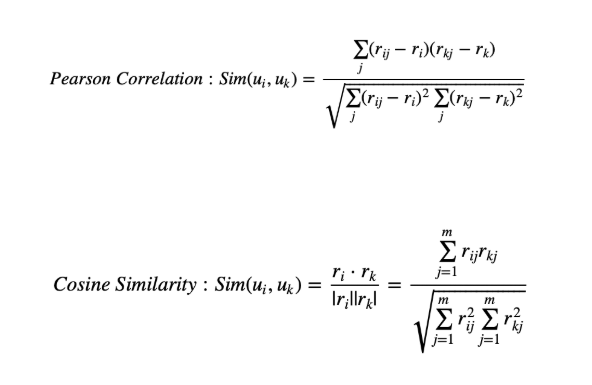


Figure Pearson Correlation & Cosine Similarity Formulas

This similarity calculation is made for finding the most similar another person to the target user. Moreover, the items rated highly are recommended to the target user.

For Item based filtering, if items received similar ratings from one user then it is assumed those items are similar. Then the recommendation can be calculated for the target user by weighted average of ratings on some number of similar items. Item based filtering has a better stability since generally rating on an item does not change frequently.

### Matrix Factorization

Collaborative filtering suffers from sparse rating matrixes and scalability for big data sets, it is more logical to decompose the sparse matrix to low dimensional matrixes with latent factors and make them denser. This process is matrix factorization.

Matrix factorization tries to group items into categories and express latent feature in higher-level attributes. Therefore, unlike the nearest neighbour, matrix factorization gives how much a user is interested in a set of latent feature and how much another item fits into this latent feature group.

Matrix factorization gives the advantage to match two users even though they have not rated same items so far. Because this method looks into underlying latent features.

In linear algebra, the singular value decomposition (SVD) is a factorization of a real or complex matrix that generalizes the Eigen decomposition of a square normal matrix to any m x n matrix via an extension of the polar decomposition (Singular Value Decomposition, 2020).SVD simply says that a real matrix R can be decomposed into 3 matrices U, Σ, and V. U is the user-latent matrix, V is a item-latent matrix and Σ is a diagonal matrix containing singular values of original matrix which is simply representing the importance of a feature.

Σ matrix’s values are sorted decreasing and the matrix is truncated for the k singular values so we can reconstruct the matrix as matrix A. This matrix is approximation of matrix R. The different between A and R is the error that is expected to be minimum as possible. This is exactly the thought of Principle Component Analysis.

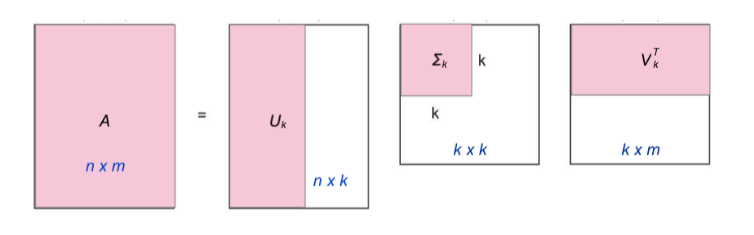


Figure Singular Value Decomposition Visualization

Factorization can be easily analysed if the starting matrix R is dense but for most cases in collaborative filtering, the R matrix generally is a sparse matrix since it is not possible to make every user rate most of the items in the data set. Instead of factorizing R with SVD another way is to find U and V matrixes directly with the goal when U and V multiplied the output matrix is R’. This R’ is the closest approximation of R and it is no more a sparse matrix. This numerical approximation is generally achieved by using Non-Negative Matrix Factorization. The reason of using non-negative factorization is that there are no any negative values in the ratings.

### Optimization

Alternative Least Square is an optimization algorithm to solve Non-Negative Factorization. Since the loss function is non-convex in this case, there is no way to reach a global minimum, while it still can reach a great approximation by finding local minimums. Alternative Least Square is to hold user factor matrix constant, adjust item factor matrix by taking derivatives of loss function and setting it equal to zero, and then set item factor matrix constant while adjusting user factor matrix. Repeat the process by switching and adjusting matrices back and forth until convergence.

## Content Based Filtering

Content based filtering is a popular choice in recommender systems since it is relatively simpler than the collaborative filtering. Content based filtering is focusing on the attributes of the items. Idea of the content based filtering is that if a user likes an item another item that is similar to this one is recommended to the user. Content based filtering mostly used in e-commerce to recommend similar items to the users who are browsing a type of item. Content based filtering can use any algorithm to find the similarities between the contents but mostly Natural Language Processing is used to extract similarities from the keywords in explanation paragraphs of the items.

The steps in recommending products or contents to the user in content based filtering are as follows:

Identify the factors which describe and differentiate the products and the factors which might be influential in weather a user would buy the product or not,

Represent all the products in terms of those factors or descriptors or attributes,

Create a tuple or number vector for each product that represents the strength of each factors for the product,

Now start to look at the users and their history and create a user profile based on their history. It will have the same number of factors and their strength would indicate how much influenced the user is towards that factor,

Recommend the user those products that are nearest to them in terms of those factors (Poudyal, 2020).

## Application Programming Interface

Applications programing interface (API) is an interface to allow interact with software through requests. The reason to implement an API, to simplify the programming by hiding unnecessary objects/code implementation from the client-side. Generally, APIs are related to software libraries and they describe the behaviour of the library instead of the actual library. An API can have more than one implementation so different libraries share the same API. There are four main types of APIs, open APIs or public APIs these type of APIs have no access restriction. Partner APIs, this type of APIs require having specific rights/licenses in order to access them. Internal APIs or private APIs, these type of APIs generally created to use within a company. The last, Composite APIs, this type of API is the combination of different data and APIs it is a sequence of tasks executed synchronously to speed up process of execution (Types of APIs, 2020). Remote APIs, Operating System APIs, Web APIs are the most known API examples. Operating System APIs lets an application to interact with the operating system. For example, Linux and Berkeley Software Distributions are example of operating systems that implement POSIX APIs. POSIX API API), along with command line shells and utility interfaces, for software compatibility with variants of UNIX and other operating systems (POSIX, 2020).

### WEB API

Web APIs are the most popular and frequently used type of APIs. The number of web APIs are getting more and more over the past years. This type of API can be for web server or web browser. Server side web API has a one or more public endpoints. Web APIs take requests as inputs and sends responses as output. Typically, any extra data desired to be sent or received are expressed in Json or Xml and request/response is carried over HTTP. There are four major web API services, Simple Object Access Protocol (SOAP), this protocol uses XML to carry data and the main function is defining messages and methods of communication (Types of APIs, 2020). XML-RPC, this protocol uses XML to carry data and uses minimum bandwidth than SOAP. Json-RPC, this protocol is using Json to carry data and is similar to XML-RPC. Representational State Transfer (REST), this is not a protocol but it is a set of principles. REST needs simple interfaces that must be satisfied there are six of them, client-server, stateless, cacheable, uniform interface, layered system, code on demand (What is REST, 2020).

# Similar Works

## Designing Activity-aware Recommender Systems for Operating Rooms

This thesis is published by Masoud Sattari in Department of Computer Engineering METU in September 2013.

This paper explains the spread of mobile phones and satellite positioning systems. With this spread, new demands for applications are emerging. Users are highly interested in location based activity recommendation.

The system that has been aimed to develop in this work is, to enhance the accuracy of hybrid systems which are created to eliminate the drawbacks of singular recommendation systems.

Hybrid systems are generally presented in two dimensional location-activity rating matrix. This thesis instead uses extra data of Singular Value Decomposition (SVD) method. This method has the functionality of uncovering latent relation within data and reducing its rank. (Sattari, 2013).

In addition, instead of 2-D rating matrix, this thesis uses 3-D user-location-activity rating matrix. This paper accomplishes this reduction by means of High Order Singular Value Decomposition as well as merging various 2-D matrices to construct an integrated matrix.

DESIGN OF PHYSICAL ACTIVITY RECOMMENDATION

SYSTEM

DESIGN OF PHYSICAL ACTIVITY RECOMMENDATION

SYSTEM

## Design of Physical Activity Recommendation System

This thesis is published by Ashkan Sami, Ryoichi Nagatomi, Kazuo Hashimoto and Masahiro Terabe in Graduate School of Biomedical Engineering Tohoku University.

This paper explains that to prevent diseases people should do physical activities however half of the people who start doing physical activities quit in six months. Since culture has a great impact on individuals this paper indicates that this study is done for Japan only. This thesis aims to prevent the health diseases that people are suffering from lack of exercise. So, a recommender system to recommend physical activities for adults has been presented. In addition, based on expertise and detailed information of different sports and activities, ontology trees and tables are built for different attributes of physical activities (Hashimoto, 2008). With these ontologies, distances of different physical activities are calculated.

# Solution Development

For developing an activity/hobby recommender system as a normal program or an application programming interface the first programming languages comes to mind is Python and so in this article the recommender system will be built in Python language with using the PyCharm IDE developed by JetBrains. The reason Python is becoming a popular choice for recommender systems or more generally artificial intelligence programs is that it has numerous libraries that support to do the artificial intelligence, deep-learning and machine learning processes. For the API implementation Flask will be used due it is the most simple web frameworks of the Python language.

Content based filtering or collaborative filtering can be used to recommend activities and they would pretty much give similar accuracy for well-defined data set. For this article, hybrid system, which consist of both collaborative filtering and content based filtering is chosen. Hybrid systems generally eliminates the disadvantages of using a single filtering method thus leads to better accuracy.

Before starting the the code the required libraries must be installed and imported. The details of these libraries can be found in Section Experimental or simply the Figure \*\* can be refered to see the name of the libraries.

First of all we will import and install the libraries and we start with initializing a new flask app. Then we define a route that will run the recommender system once it is requested over HTTP. For this example in order to focus on the artificial intelligence rather than web development we will choose the most simple route “/”.



Figure Importing Libraries and Creating Flask Application and Defining a Route

In order to give recommendations to a user, we must know at least one item that has already been rated/liked by the user so for this this program takes a single activity from the user to recommend similar activities.

From this part, the collaborative filtering code starts. We read two CSV files which are data set of this project. With the Pandas library CSV files are loaded as dataframes to our program.



Figure Reading CSV Formatted Files and Loading Them as Pandas Dataframes

Once dataframes are ready, now we will merge these two dataframes by their common column, activityId. This will produce a new dataframe where we can see every rating for each activity. Once the merged dataframe is ready, now we can calculate the average rating of each activity and the number of ratings it has got by using groupby() operation on the columns “title” and “rating”.

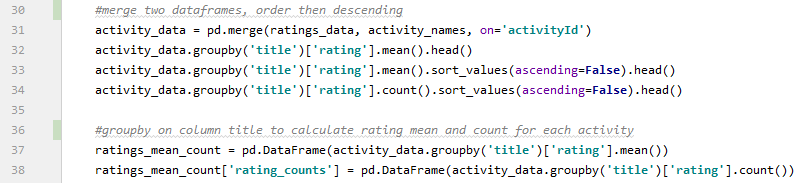


Figure Merging Dataframe on "activityId" and Calculating Mean Ratings & Rating Counts

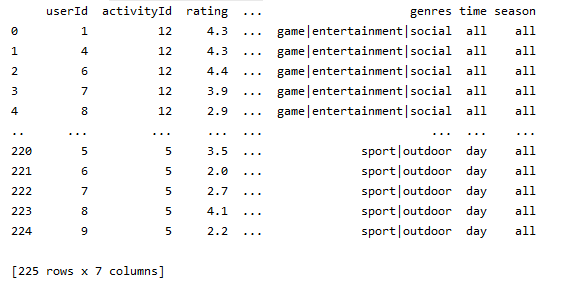


Figure Result Dataframe after “ratings\_data” & “activity\_names” Dataframes are Merged on “activityId”

Now we will use Matplotlib’s Pylplot to visualize the ratings and counts in histograms. This step is optimal but it is recommend to analyse the dataset’s quality.

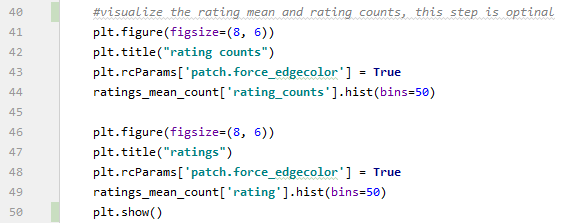


Figure Plotting Two Histograms in Seperate Matplotlib Figures

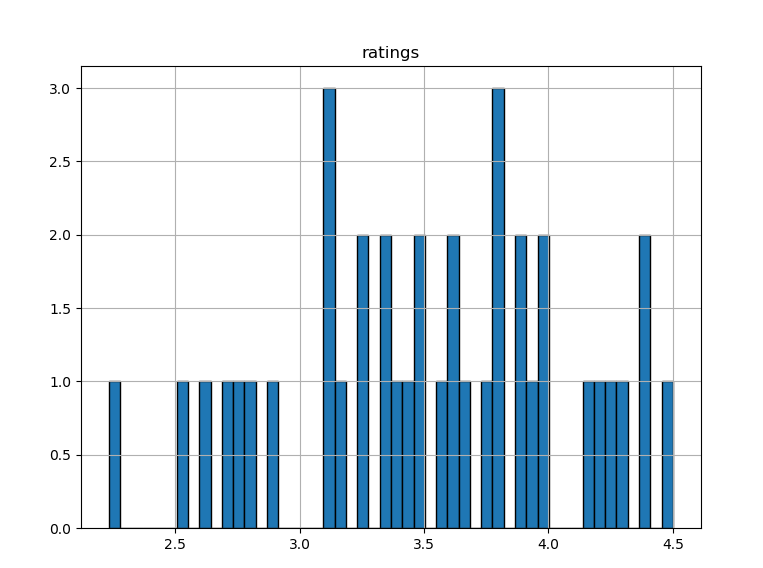


Figure Histogram that Shows Spread of Ratings

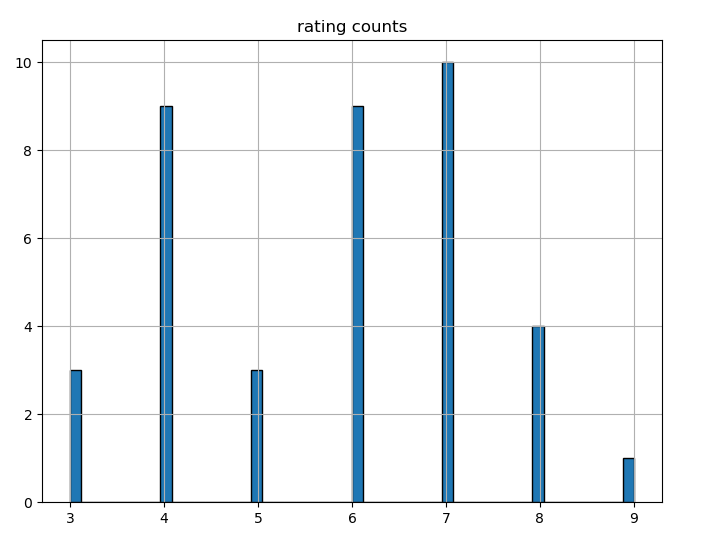


Figure Histogram that Shows Spread of Rating Counts

As we can see in the ratings histogram, the ratings CSV file is pretty much balanced. For the rating counts we can understand that the maximum number of ratings to a specific activity is 9 so this indicates that the ratings CSV file does not contain many users but still good enough to make recommendations.

After analysing the data we have, now we find the correlation between users and activity, we need to form the rating matrix of user-activity.

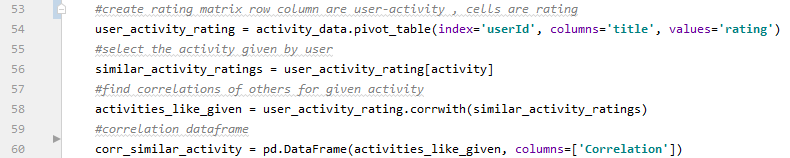


Figure Creating Rating Matrix and Calculating Pairwise Correlation

After we successfully calculate correlations now in order to tune the results we should eliminate the null cells from the rating matrix and put a threshold of minimum rating count to eliminate subjective ratings.

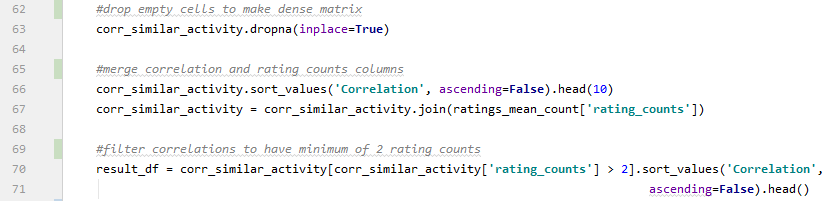


Figure Dropping Empty Cells on Ratings Matrix and Filtering Correlations with Minimum 2 Rating Counts

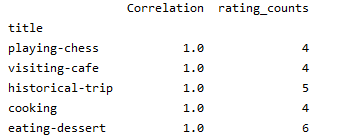


Figure Correlation Dataframe after It Undergoes Operations in Figure 12

Even though we have improved the results by giving a minimum number of rating counts limit and removing null cells from the matrix, we still need to filter the result so the user can do any of the recommendation at that moment. For example recommending a user to go swimming in winter is pointless.

For eliminating the irrelevant recommendations, we implement two supportive methods, which compare the current date time with the activity’s recommended time & season data from the dataframe.

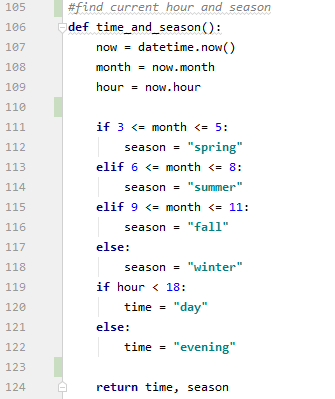


Figure Python Method that Finds the Current Hour of Day and Season

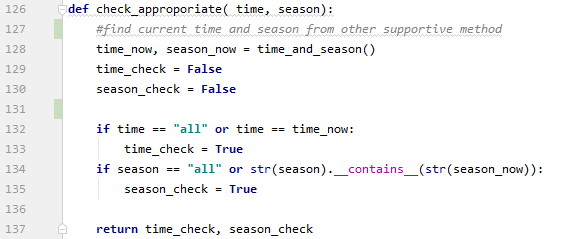


Figure Python Method that Checks whether the Activity Fits in with Current Hour and Season

Now we will call these supportive methods to filter the activitiy(s) according to the hour of the day and the current season.

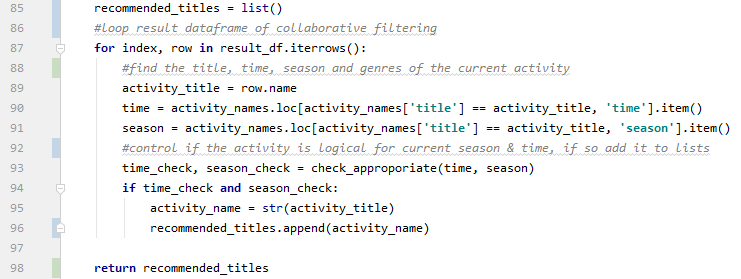


Figure Loop which Calls Supportive Methods for each Activity

To this point, we have implemented the collaborative filtering with pairwise correlation. Filtered the result of collaborative filtering according to season & time and recommendations list can be used to finish the recommender system but in order to increase the accuracy of recommendations we can send the result of collaborative filtering to the content based filtering and complete the hybrid system.

Before the result is sent to the content based filtering, we need a little adjustment to do. We need to create another list to hold the genres of the activities that are inside recommendations list. Because content based filtering will use these genres to find similarity from keyword matching which is a natural language processing method.

We add title and genres of the activity that is given by the user to the appropriate lists and call the method that will do the content based filtering. Now the code from the figure above looks like this.

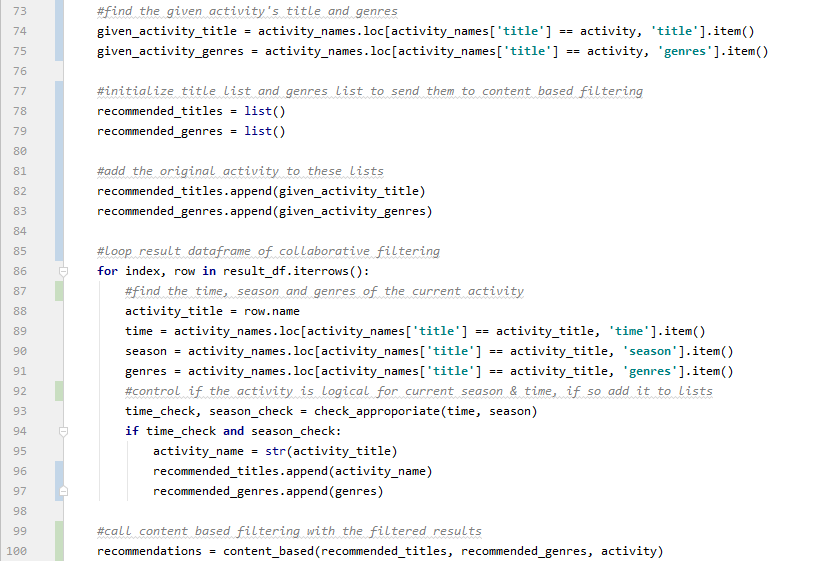


Figure Python Code that has Changed to Implement Hybrid System

Content based filtering will take the given title and genres list and will create a new dataframe from them for extra an empty column named “Key\_words” is added to this dataframe to hold the extracted keywords that will be extracted in the next step.

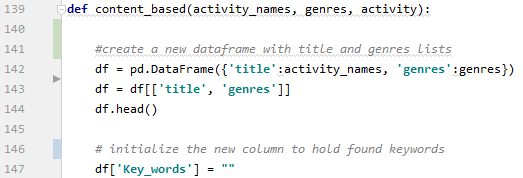


Figure Creating a New Dataframe for Content Based Filtering

Now, we loop all the rows of the dataframe which is equivilant to the filtered result of collaborative filtering plus the original activity’s data. For each activity, we create a new vectorizer and extract the keywords and assigning the result dictionary to the “Key\_words” column of the activity. This process is a simple NLP method to find similarities of items by their description paragraphs, which is genres column for our dataset.

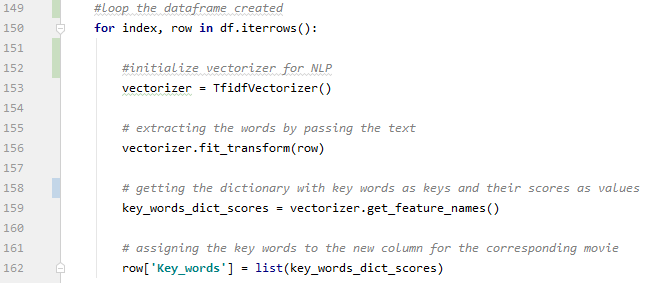


Figure Extracting Keywords of each Activity in Dataframe

Now we create count matrix to calculate how many times a keyword repeated for each activity. With the count matrix we can generate a cosine similarity matrix to calculate/find the activity that share common keywords.

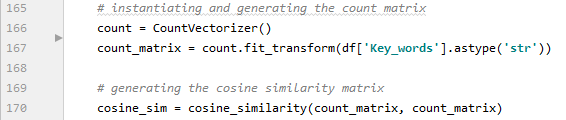


Figure Initializing Count Matrix to Calculate Cosine Similarity

We find the index of the original activity given by the user to create a Series and sort the dateframe by the cosine similarity value in descending order. Then we simply take the first 5 rows of this Series which is equivalent to the most similar 5 activities. Before giving the recommendations to the user we remove the original activity from the result list to not recommend what user already have given us.

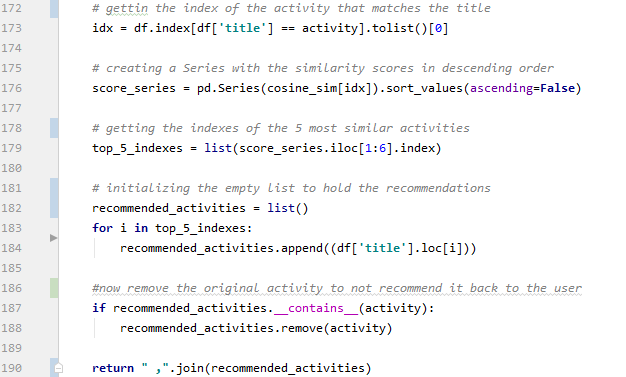


Figure Sorting the Count Matrix and Selecting Top 5 Activities

After the recommendation list is processed the method returns back to the collaborative filtering and there is nothing left to do but giving the recommendations to the user. As it was indicated before, the API is kept as simple as possible to without a further work we simply put the result inside header element and present it to the user.

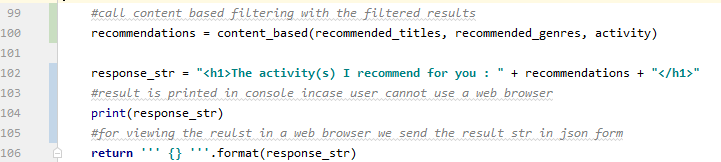


Figure The End of the Collaborative Filtering Method

This program is ran just same as a Python console application and the result of the execution can be examined in either console or web browser. In order to start execution of the filtering we need to go to the app.route we have defined which can be found in the console once the Flask application is ran.

To run the Python Flask application simply right click on the code and select the option called “Run NameOfCode.py”.

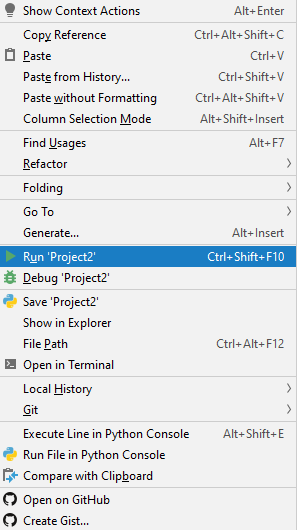


Figure Options of Right Cliking on Code Page

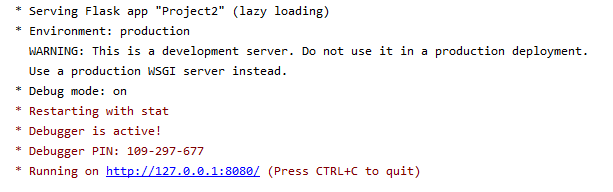


Figure Console Output of Running the Flask Application

Clicking and going to the local link provided in the console directly executes the recommender system and its result can be viewed.

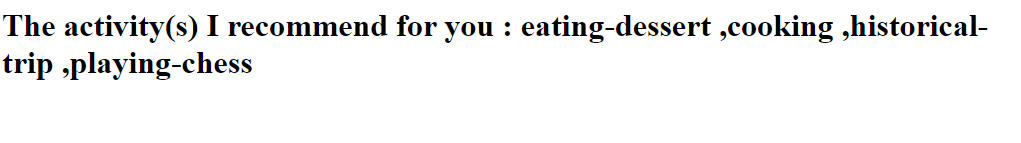


Figure Visualising of Recommender System’s Result in Web Browser

# Experimental Work

## Dataset

For implementing a recommender uses collaborative filtering, data set must be researched/prepared. For this article and recommender system there was no any activity/hobby dataset prepared for collaborative filtering. Therefore, a custom data set has been prepared from scratch just for this article.

To supply the recommender the activities and their attributes a CSV file named “Activities” is prepared. This dataset contains 40 rows/activities and 5 columns which are, activity id, title, genre, time and season.

* Activity id: Unique integer number identifying the activity.
* Title: Name of the activity, if an activity name consists of multiple words they are binded with “-“ character.
* Genres: Types of the activity to describe it in details, different genres are split with “|” character.
* Time: Time of the activity can be done. There are day, evening and all to represent time. Day represents 8 A.M to 6 P.M, evening represents 6 P.M to 8 A.M. and “all” for any hour of the day.
* Season: Season of the activity can be done. Can be any combination of the four seasons, split with “-“ character if multiple. Alternatively, can be “all” which represents all the of year.

There is also a second CSV file which represents the ratings given to activities by users named ”Activity Ratings”. This file is currently containing ratings randomly created so it lacks the logical sense. There are total of 10 users who have rated 25 random activities with random points in between 0.1 to 5.0. This data set contains 225 rows and 3 columns which are user id, activity id and rating.

* Activity id: Unique integer number identifying the activity.
* User id: Unique integer number identifying the user.
* Rating: A double number in between 0.0 to 5.0 indicating the point given to an activity. Higher the numbers indicates the activity is liked more.

## Libraries

For implementing this Recommender System there – number of libraries will be needed. These libraries all are useful to implement an aspect of the recommender system.

Pandas: Pandas is an open sourced library is allowing us to create DataFrames out of tabular data such as the CSV data set files or spreadsheets. Pandas allows statistical operations to be calculated on these DataFrames.

Pandas 0.25.3 Version is used in this article and it can be installed via the commands “conda install pandas” or “pip install pandas”.

Matplotlib: Matplotlib module is allowing us to create visualization of data. It also allows user to draw graphs without needing of a prior data. Matplotlib creates Figure Windows to present the visuals. In this recommender system Matplotlib module is optional to install, we will only use this to visualize the rating and rating counts of the data sets.

Matplotlib 3.1.1 Version is used in this article and it can be installed via the command “python –m pip install –U matplotlib”.

Datetime: Datetime module is a default module comes with Python. We will use this module to get the current date and time to control if an activity is fit to the current situations.

Flask: Flask web framework is used to implement an API. Flask is used in this article to create web application. Recommender system is implemented as an API but readers can use the code and logic explained in the article to implement the system as a console application too.

Flask 1.1.1 Version is used and it can be installed via command “pip install –U Flask”.

Sklearn: Sklearn is a machine learning module which offers numerous machine learning algorithms. Module also supports numerical and scientific modules NumPy and SciPy inside it. Sklearn module is used in this article to calculate the complex keyword extraction and cosine similarity operations and iniatlizing Count Matrix.

Sklearn 0.23.0 Version is used in this article and it can be installed via commands “pip install –U scikit-learn” or “conda install scikit-learn”.

## Improving Results

For this recommender system, “Solution Development” chapter is explained by using the best resulted dataset and filtering techniques. This chapter will focus on the preliminary versions of the recommender system where there were less data or no filtering of the results.

At first, the dataset was containing only 4 different users who have rated only 10 random activities among the total of 40 activities and the minimum number of rating counts limit is given as 2 to the collaborative filtering. Assuming the user has given “visiting-café” activity the result of the recommender system was very poor. It could only recommend two activities



Figure Recommendation with Less Ratings & Threshold Set to Two

With containing the low population dataset and increasing the threshold to the 7, as easily predicted gave no any recommendations because there were no any activities that ever had more than 7 ratings.

With the data set has been populated to the 10 users ratings 25 activities. The collaborative filtering gives more activities as recommendation.

With increasing the ratings dataset population to 10 different users rating 25 random activities. When the recommender system executed with threshold of two, recommender could recommend more activities and the correlation of these activities were almost 1.0 for all of it. Even though correlations were high, the results were not satisfying.

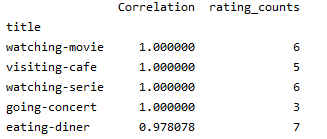


Figure Result with More Ratings & Threshold Set to Two

Now with containing the crowded data set but increasing the threshold to seven the number of recommendations has kept the same (The visiting-café of the Figure is not counted since it is exactly same with the user has already given). This time correlations have dropped very low but the recommendations given were very satisfactory.

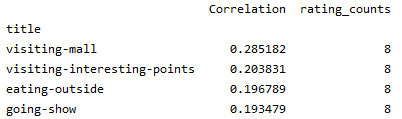


Figure Result with More Ratings & Threshold Set to Seven

Some experimental changes has been done to the code and the result of recommendation is analysed. Increasing the rating population helps to give more recommendations. When the minimum rating count threshold is increased the recommendation set changes compared to the lower threshold set. Even though the average correlation decreases when the threshold is increased recommendations get more logical.

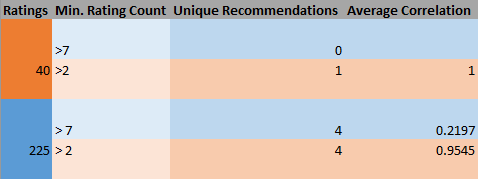


Figure Results with Different Ratings Amount and Thresholds

# References

*Artificial Intelligence*. (2020, May). Retrieved from Wikipedia: http://wiki.pinsify.xyz/index.php?q=aHR0cHM6Ly9lbi53aWtpcGVkaWEub3JnL3dpa2kvQXJ0aWZpY2lhbF9pbnRlbGxpZ2VuY2U

Hashimoto, A. S. (2008, January). *Design of Physical Activity Recommendation System.* Retrieved from Research Gate: https://www.researchgate.net/publication/220970088\_Design\_of\_Physical\_Activity\_Recommendation\_System

*Introduction to Recommender System*. (2020, May). Retrieved from Towards Data Science: https://towardsdatascience.com/intro-to-recommender-system-collaborative-filtering-64a238194a26

*POSIX*. (2020, May). Retrieved from Wikipedia: http://wiki.pinsify.xyz/index.php?q=aHR0cHM6Ly9lbi53aWtpcGVkaWEub3JnL3dpa2kvUE9TSVg

Poudyal, R. (2020, May). *Content Based Filtering in Recommendation Systems*. Retrieved from Medium: https://medium.com/@rabinpoudyal1995/content-based-filtering-in-recommendation-systems-8397a52025f0

*Recommmender System*. (2020, May). Retrieved from Wikipedia: http://wiki.pinsify.xyz/index.php?q=aHR0cHM6Ly9lbi53aWtpcGVkaWEub3JnL3dpa2kvUmVjb21tZW5kZXJfc3lzdGVt

Sattari, M. (2013, September). *A HYBRID GEO-ACTIVITY RECOMMENDATION SYSTEM USING ADVANCED FEATURE COMBINATION AND SEMANTIC ACTIVITY SIMILARITY.* Retrieved from Middle East Technical University: http://etd.lib.metu.edu.tr/upload/12616497/index.pdf

*Singular Value Decomposition*. (2020, May). Retrieved from Wikipedia: http://wiki.pinsify.xyz/index.php?q=aHR0cHM6Ly9lbi53aWtpcGVkaWEub3JnL3dpa2kvU2luZ3VsYXJfdmFsdWVfZGVjb21wb3NpdGlvbg

*Types of APIs*. (2020, May). Retrieved from RapidApi: https://rapidapi.com/blog/types-of-apis/

*What is REST*. (2020, May). Retrieved from RestApi: https://restfulapi.net/