EGE UNIVERSITY

COMPUTER ENGINEERING DEPARTMENT

ARTIFICIAL INTELLIGENCE

&

DEEP LEARNING

PROJECT 2

Deniz YÜREKDELER - 05180000017

Delivery Date

21.05.2020

Introduction

With the growth of mobile phones and internet, 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. 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. 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 system 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 for content based filtering's keyword extraction. This recommender system will be written in Python's Flask web framework 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 keeping the focus on the artificial intelligence.

Artificial Intelligence

Artificial Intelligence (AI) can be simplified as, a machine that is aware of its environment to take proper actions to reach its goal successfully. AI is a broad concept, though it is mostly referred to computers mimicking human learning. Modern machine capabilities generally classified as AI include successfully understanding human speech, competing at the highest level in strategic game systems, autonomously operating cars, intelligent routing in content delivery networks, and military simulations (Artificial Intelligence, 2020).

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 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

Collaborative Filtering (CF) finds similarities of user behaviours to recommend them some item that the other liked. CF can be generally divided in user-based and item-based filtering. 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. CF uses different algorithms to make the prediction, which is a complicated statistical calculation. There are Nearest Neighbourhood, Matrix Factorization and Optimization methods for the prediction.

Content Based 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 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 explanations.

Application Programming Interface

An application programming interface (API) is a set of definitions and protocols for building and integrating application software (What is an API?, 2020). The reason to implement an API, to simplify the programming by hiding unnecessary objects/code implementation from the client-side. Web APIs are the most popular and frequently used type of APIs. 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. Request/response is carried over HTTP.

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

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.

Difference from Similar Works (Extra 2)

Compared to examples in literature, they generally use singular recommender systems; instead, this article develops a hybrid system. In addition, other works generally are implemented in one platform but this article's recommender is implemented as a web API form, thus allowing it to be used platform free. In addition, unlike other recommenders of location based activity or physical exercises, this system offers to recommend hobbies to users as extra.

Solution Development

For developing an activity/hobby recommender system as an API, the first programming languages comes to mind is Python. In this article, the recommender system will be built in Python 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 artificial intelligence, deep-learning and machine learning algorithms. For the API implementation, Flask web framework of Python will be used due it is the one of the most simple web frameworks.

Solution Pseudocode

Read Dataset to dataframes and inner join (merge) tables
Group by on common column and find mean & count of each element
Create rating matrix with user-element and cells are rating
Eliminate null cells and but threshold to count in rating matrix
Calculate person correlation with given activity
Filter results according to the season and hour
Loop each row of the result, find keywords and append dataframe with keywords
Create count matrix and calculate cosine similarity
Find index of given activity and sort cosine similarity results descending by the index found
Present user the top five elements of the sorted list

For this article, a hybrid system, which consist of both collaborative filtering and content based will be implemented. Hybrid systems generally eliminates the disadvantages of using a single filtering method thus leads to better accuracy. Before starting the code, the required libraries must be installed and imported. The details of these libraries can be found in Experimental Work section.

In order to give recommendations to a user, we must know at least one item that has already been liked by the user. This program takes a single activity from the user to recommend other similar activities. In our hybrid system, we start with the collaborative filtering and read two CSV files/data to Pandas dataframes. We will merge these two dataframes by their common column, activityId. This will produce a new dataframe that can be seen in Figure 1. 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”. In addition, we can use Matplotlib’s Pylplot to visualize the ratings and counts in histograms.

	userId	activityId	rating	...	genres	time	season
0	1	12	4.3	...	game|entertainment|social	all	all
1	4	12	4.3	...	game|entertainment|social	all	all
2	6	12	4.4	...	game|entertainment|social	all	all
3	7	12	3.9	...	game|entertainment|social	all	all
4	8	12	2.9	...	game|entertainment|social	all	all
..	...	...	...	...	...	...	...
220	5	5	3.5	...	sport|outdoor	day	all
221	6	5	2.0	...	sport|outdoor	day	all
222	7	5	2.7	...	sport|outdoor	day	all
223	8	5	4.1	...	sport|outdoor	day	all
224	9	5	2.2	...	sport|outdoor	day	all

[225 rows x 7 columns]

Figure 1 Result Dataframe After “ratings_data” & “activity_names” Dataframes are Merged on “activityId”

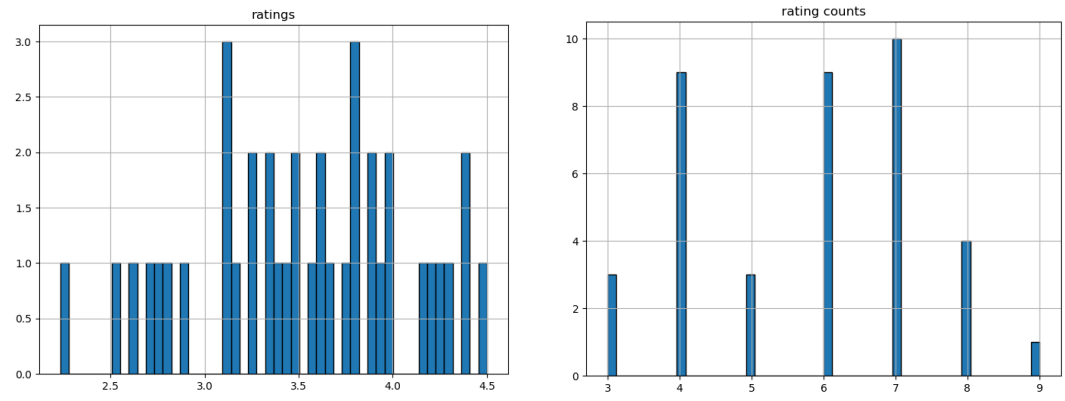

Figure 2 Histogram that Shows Spread of Rating Counts

As we can see in the ratings histogram, the ratings CSV file is pretty much balanced. For the rating counts, we can understand it does not contain many users but still good enough to make recommendations. We create a "rating matrix" and calculate Pearson correlation. Null cells are eliminated from rating matrix and a threshold has been given to eliminate subjective ratings.

title	Correlation	rating_counts
playing-chess	1.0	4
visiting-cafe	1.0	4
historical-trip	1.0	5
cooking	1.0	4
eating-dessert	1.0	6

Figure 3 Correlation Dataframe after It Undergoes Operations

After recommendations found we filter them by season and hour. For example recommending a user to go swimming in winter is pointless. So far, we have implemented the collaborative filtering with Pearson correlation. Now we send the result of collaborative filtering to the content based filtering for implementing hybrid system. Content based filtering will create a new dataframe from given result and loop all the rows of the dataframe to create a new vectorizer and extract the keywords of genres column. Keywords found are appended to the dataframe. Once keywords of all activities are found, similarities are calculated with count matrix and cosine similarty.

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 five rows of this Series which is equivalent to the most similar 5 activities and present it to the user in header element in a web browser.

Guides Followed and Differences (Extra 3)

A guide for Collaborative Filtering (Creating a Simple Recommender System in Python using Pandas, 2020) is followed and it is improved by using custom dataset with filtering the recommendations depending on the season and hour. Another guide for Content Based Filtering (How to build a content-based movie recommender system with Natural Language Processing, 2020) is followed and these two recommender systems are mixed to implement a hybrid system. Content based filtering had major changes on NLP algorithm for keyword extraction (Guide was using RLTK to keyword extraction, this article uses Sklearn). Data manipulation processes are added as extra for implementing the hybrid system and presenting results to the user, these were missing in the guides followed.

Experimental Work

Dataset

For this article and recommender system, custom data set has been prepared from scratch. “Activities” CSV file contains 40 rows/activities and five columns which are, activity id, title, genre, time and season. There is also a second CSV file named ”Activity Ratings”, which contains the ratings given to activities by users. This file is currently containing ratings randomly created, so it lacks the logical sense. This data set contains 225 rows and 3 columns, which are user id, activity id and rating.

Libraries

Pandas (1.0.3) allows creating DataFrames from dataset. Install with “pip install pandas”. Matplotlib (3.2.1) allows visualizing data. Install with “python –m pip install –U matplotlib”. Flask web framework (1.1.2) allows implementing an API. Install with “pip install –U Flask”. Sklearn (0.23.1) is a machine-learning module contains algorithms. Install with “pip install –U scikit-learn”.

Improving Results (Extra 1)

In order to improve the results, minimum number of rating counts limit added before calculating correlation and null cells removed from the rating matrix. Result of collaborative filtering is filtered by the current season and hour. With these three changes, recommendations were more acceptable by the user. Apart from the changes in the code, dataset has changed for improving result.

At first, the rating dataset was containing 40 ratings; only 4 different users who have rated 10 random activities among the total of 40 activities and the minimum number of rating counts limit was given as 2. Assuming the user has given “going-cinema” activity, the result of the recommender system was very poor. It could not recommend anything.

With increasing the number of ratings to 225, dataset population to 10 different users who are rating 25 random activities, recommender finally could give some recommendations. With populated dataset and increasing the threshold to 5 or 7, the recommendations became much more logical compared to threshold 2. However, when the threshold was too high, recommender was giving same recommendations for different activities that are given by the user. The reason for that, there were only a few activities rated more than 7. Setting the threshold to 5 was the best result since there were plenty of activities with more than 5 rating counts. With the threshold of 5, recommendations were highly logical and changing with each given different activity.

title	Correlation	rating_counts
visiting-mall	0.285182	8
visiting-interesting-points	0.203831	8
eating-outside	0.196789	8
going-show	0.193479	8

Figure 4 Result with More Ratings & Threshold Set to Seven

Ratings	Min. Rating Count	Num. Of Unique Reco.	Average Correlation	Num. Of Reco. Actually Presented	Logic	Does Reco. Change
225	> 2	4	0.99	3	Medium	Yes
	> 5	5	0.92	2	High	Yes
	> 7	3	0.4	1	High	No

Figure 5 Experimental Analysis of Populated Dataset with Different Thresholds

Future Work

Recommender system will be changed to accept more than one given activity. Ratings dataset will be prepared with actual user data instead of using random and changes on result will be compared. For experimental purposes, order of the recommender methods will be swapped to see if the results get better.

Conclusion

Increasing the rating population helps to give more recommendations. When the minimum rating count threshold is increased, recommendations get more logical but if it is too high recommendations to different inputs does not change. With a dense rating matrix, results are better compared to sparse rating matrix. More the genres column get details, better the content-based filtering recommends because it can find distinctions easily. Setting threshold to more moderate number increases the variation of recommendations.

Self-Assessment Table

	Desired Criteria	Done	Explanation	Estimated Grade
1	**Literature research (10)**	☒	Some similar works examined. Technical information and two of the similar works explained in simple and original sentences. Reader can easily understand the technical expressions used in this article by reading the literature section.	10
2	**Introduction, conclusion sections (10)**	☒	Introduction and conclusion expressed in a way to interest the readers just as in online guides/blogs. These sections are kept simple, original and explanatory to readers so they can understand remaing parts of the article easier or sum up the things have been done in the article.	10
3	**Problem and solution method (10)**	☒	Problem is an original thought. Solution and methods explained with giving the reasons why those solution steps were taken so readers can understand the logic behind the implementations. Source code segmented and commented for readers to easily understand which paragraph belongs to that code block.	10
4	**Experimental work & extra 1 to improve success (10)**	☒	Four different approaches experimented to increase the success. Their results explained with giving the after-before results and a table has been drawn to show change of results. Experimental works are logical and have actually increased the success rate of the recommender system.	10
5	**Article Format, Organization, Size, Quality, Source and Citations (10)**	☒	Report was written longer in original but it has been simplified to meet the page requirements. Yet still, it is kept as organized as possible with giving enough of the information. All references included in the paragraphs to show respect to the writers. Microsoft Word's recommended format used in headings and paragraphs.	10
6	**Extra 2 & extra 3 (10)**	☒	A lot of effort put in to give uniqueness to the system designed in this article. All the guides followed for implementation are referenced. Extras are placed under the approporiate headings.	10
7	**Progress report & code demo (10)**	☒	Recommender system has been tested for defects and it works as expected. By the time of submiting the progress report, the project (code&report) has been completed. Future works will be implemented until final work submission.	10
8	**Q&A in Teams with code explanation and demo (20)**	☒	This article's code and report has been written in original sentences/logics and all the ascepts required to complete this project has been researched/studied before completing it. Any question can be answered.	20
9	**Self Assessment Table (10)**	☒	Explanations of the table are kept as short as possible.	10
			Total grade out of 100:	100

Bibliography

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

Creating a Simple Recommender System in Python using Pandas. (2020, May). Retrieved from Stack Abuse: https://stackabuse.com/creating-a-simple-recommender-system-in-python-using-pandas/

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

How to build a content-based movie recommender system with Natural Language Processing. (2020, May). Retrieved from Towards Data Science: https://towardsdatascience.com/how-to-build-from-scratch-a-content-based-movie-recommender-with-natural-language-processing-25ad400eb243

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=aHR0cHM6Ly9lbi53aWtpcGVkaWEub3JnL3dpa2kvUE9TVg

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 an API? (2020, May). Retrieved from Red Hat: https://www.redhat.com/en/topics/api/what-are-application-programming-interfaces

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