

MOVIE RECOMMENDATION SYSTEM

END TERM REPORT

BY

**TUSHAR VERMA, RISHIDEEP PAN, VISHU VYAS, ABHINAV ANAND
SINGH**

(Section: K18JF)

(Roll Number: 03, 02, 04, 52)



LOVELY
PROFESSIONAL
UNIVERSITY

Department of Intelligent Systems

School of Computer Science Engineering

Lovely Professional University, Jalandhar

April-2020

Student Declaration

This is to declare that this report has been written by me/us. No part of the report is copied from other sources. All information included from other sources have been duly acknowledged. I/We aver that if any part of the report is found to be copied, I/we are shall take full responsibility for it.

Tushar verma

Roll Number: 03

Rishideep Pan

Roll Number: 02

Vishu Vyas

Roll Number: 04

Abhinav Anand Singh

Roll Number: 52

Place- Lovely Professional University

Date- 10-04-2020

BONAFIDE CERTIFICATE

Certified that this project report “MOVIE RECOMMENDATION SYSTEM” is the bonafide work of “TUSHAR VERMA, RISHIDEEP PAN, VISHU VYAS, ABHINAV ANAND SINGH” who carried out the project work under my supervision.

DIPEN SAINI

23681

Department of Intelligent Systems

Description of Project

Recommender systems are information filtering tools that aspire to predict the rating for users and items, predominantly from big data to recommend their likes. Movie recommendation systems provide a mechanism to assist users in classifying users with similar interests. This makes recommender systems essentially a central part of websites and e-commerce applications. This article focuses on the movie recommendation systems whose primary objective is to suggest a recommender system through data clustering and computational intelligence. In this research article, a novel recommender system has been discussed which makes use of k-means clustering by adopting cuckoo search optimization algorithm applied on the Movie lens dataset. Our approach has been explained systematically, and the subsequent results have been discussed. It is also compared with existing approaches, and the results have been analysed and interpreted. Evaluation metrics such as mean absolute error (MAE), standard deviation (SD), root mean square error (RMSE) and t-value for the movie recommender system delivers better results as our approach offers lesser value of the mean absolute error, standard deviation, and root mean square error. The experiment results obtained on Movie lens dataset stipulate that the proposed approach may provide high performance regarding reliability, efficiency and delivers accurate personalized movie recommendations when compared with existing methods. Our proposed system (K-mean Cuckoo) has 0.68 MAE, which is superior to existing work (0.78 MAE) and also has improvement of our previous work (0.75 MAE).

Introduction

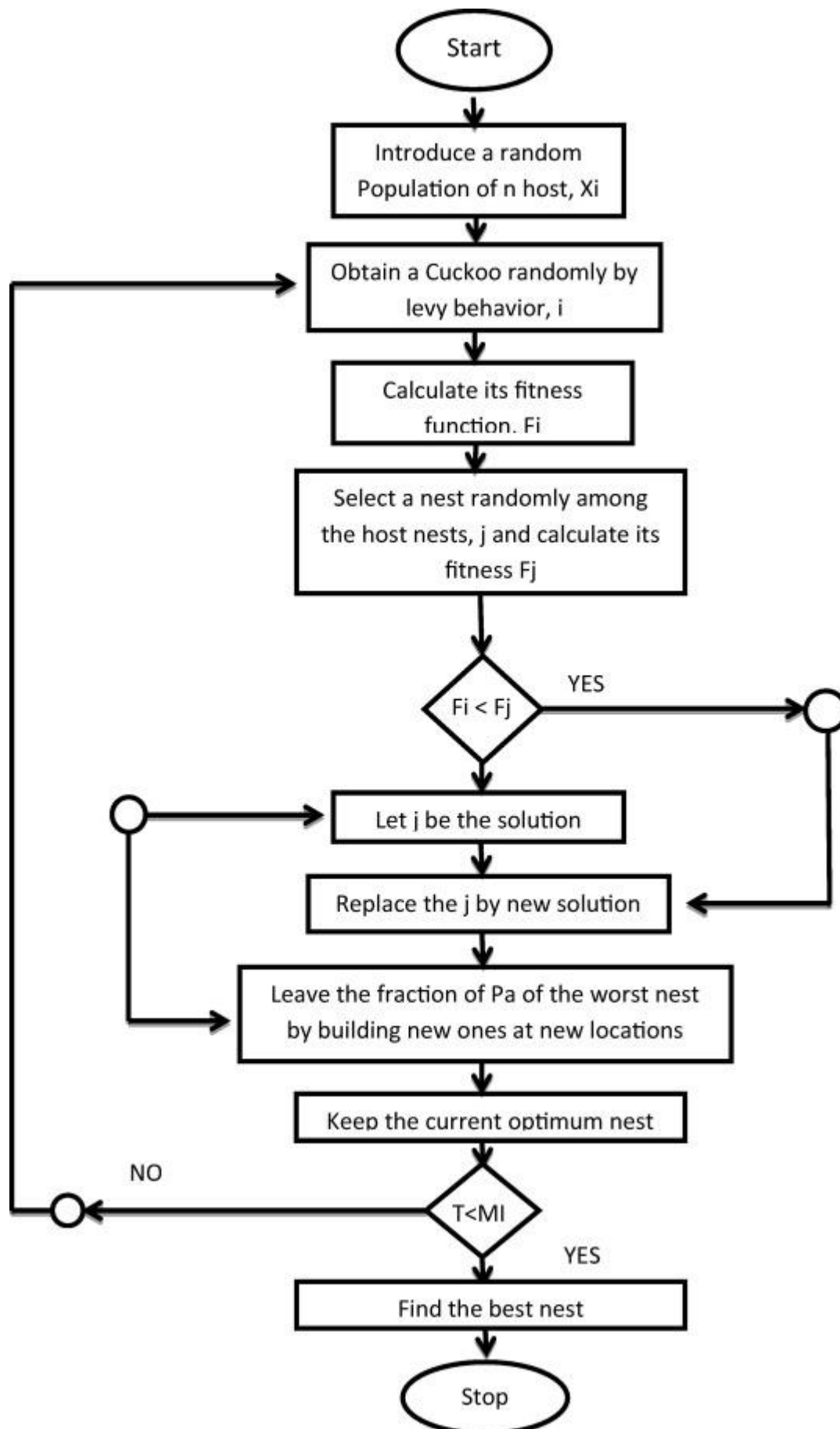
A recommendation system has become an indispensable component in various e-commerce applications. Recommender systems collect information about the user's preferences of different items (e.g. movies, shopping, tourism, TV, taxi) by two ways, either implicitly or explicitly. An implicit acquisition of user information typically involves observing the user's behaviour such as watched movies, purchased products, downloaded applications. On the other hand, a direct procurement of information typically involves collecting the user's previous ratings or history. Collaborative filtering (CF) is the way of filtering or calculating items through the sentiments of other people. It first gathers the movie ratings given by individuals and then recommends movies to the target user based on like-minded people with similar tastes and interests in the past. Additional impression on which some recommender systems are based is clustering. Clustering is a popular unsupervised data mining tool that is used for partitioning a given dataset into homogeneous groups based on some similarity or dissimilarity metric. Collaborative filtering and clustering have been discussed in detail in the next section. Hybrid cluster and optimization approach is applied to improve movie prediction accuracy. Such a hybrid approach has been used to overcome the limitations of typical content-based and collaborative recommender systems. For clustering, k-means algorithm is applied and for optimization, cuckoo search optimization is implemented. K-means algorithm is an enormously greater clustering algorithm when compared to other clustering methods in relations of time, complexity or effectiveness for a particular number of clusters. Clustering algorithm with a bio-inspired algorithm such as cuckoo search delivers optimize results. The cuckoo search has shown best performance when compared with other algorithms such as genetic algorithms and particle swarm

optimization. Simulations and comparison of the cuckoo search were greater to these existing algorithms for multimodal objective functions. To find the best results we have to find the most suitable weight among all possible ones. Cuckoo search was also performed well and showed good results that found the appropriate weights. That is why cuckoo optimization algorithm is also used to obtain optimized weight in our work. Besides being one of the most efficient algorithms, it was found that it takes less time than other algorithms applied to the same dataset. The approach of k-means and cuckoo has been applied to the dataset, and the results have been observed regarding evaluation metrics such as mean absolute error (MAE), standard deviation (SD), root mean square error (RMSE) and t-value. These parameters examined and discussed to evaluate the performance of movie recommendation system. Regarding accuracy and precision, the experiment results reflect that the proposed approach is capable of providing more reliable movie recommendations as compared to the existing cluster-based CF methods. In numerous researches, the clustering approaches are conducted with the entire dimensions of data which might lead to somewhat inaccuracy and results in more computation time. In general, designing expert movie recommendations is still a challenge, and discovering effective clustering method is a critical problem in this condition. To address aforementioned, a hybrid model- based movie recommendation approach is proposed to alleviate the issues of both extraordinary dimensionality and data sparsity. That is the reason we selected a cuckoo algorithm with k-means for optimization. On the comparison with some other optimization algorithms, the cuckoo was found to perform better than others. The major contributions of this research work are:

- We proposed a novel recommender system with K-means & cuckoo search optimization.
- Our system is innovative and efficient so far, as it employed Cuckoo search algorithm for excellent recommendations for Movie lens Dataset.
- Our hybrid model has 0.68 MAE, which is superior to existing work (0.78 MAE).
- Our model also has excellent improvement of our previous work (0.75 MAE).
- The performance with respect to time is also better as compared to already existing systems.
- We used well known Movie lens dataset (<http://grouplens.org/datasets/movielens/100k/>) to analyse the behaviour of our proposed system.

The remainder of this article is planned as follows: Section 2 gives a brief explanation of the related work that was carried out on collaborative recommendation systems and clustering-based collaborative recommendation. The proposed approach called as a k-mean-cuckoo approach for movie recommender system is explained in Section 3. In Section 4, experiment results performed on Movie lens dataset are

described, and finally summarization of this article with future work are highlighted in Section 5.



Contribution

1. Tushar Verma – Built the whole project and error tester of the project. Make the synopsis. Make the whole report of the project.
2. Rishideep Pan- Help in finding details of the project.
3. Vishu Vyas- Help in finding the data and details about the synopsis.
4. Abhinav Anand Singh- Help in finding data of the project.

Technologies and Framework

- To built this project we have use Python programming.
- Also uses module like Beautiful soup.
- Fetch data from urllib module.
- Taking the information about movies from imdb module.

SWOT Analysis

- ❖ **Strength-** Program is compact and flexible which takes less time to execute.
- ❖ **Weakness-** As it takes the data from the Internet, there is a risk of Data collapse.
- ❖ **Opportunities-** Others will gets a good idea about new and advance strategies from this.
- ❖ **Threats-** As we are fetching the data from internet so, there is a risk of hacking and changing the data.

Program

```
from bs4 import BeautifulSoup as SOUP                # Import library for web
import re
import requests as HTTP

def main(emotion):

    # movie against emotion Sad
    if(emotion == "Sad"):
        urlhere='http://www.imdb.com/search/title?genres=drama&title_type=feature&sort=
        moviemeter, asc'

    # movie against emotion Disgust
    elif(emotion == "Disgust"):
        urlhere='http://www.imdb.com/search/title?genres=musical&title_type=feature&sort=
        moviemeter, asc'

    # movie against emotion Anger
    elif(emotion == "Anger"):
        urlhere='http://www.imdb.com/search/title?genres=family&title_type=feature&sort=
        moviemeter, asc'

    # movie against emotion Anticipation
    elif(emotion == "Anticipation"):
        urlhere='http://www.imdb.com/search/title?genres=thriller&title_type=feature&sort=
        moviemeter, asc'

    # movie against emotion Fear
    elif(emotion == "Fear"):
        urlhere='http://www.imdb.com/search/title?genres=sport&title_type=feature&sort=m
        oviemeter, asc'

    # movie against emotion Enjoyment
    elif(emotion == "Enjoyment"):
        urlhere='http://www.imdb.com/search/title?genres=thriller&title_type=feature&sort=
        moviemeter, asc'

    # movie against emotion Trust
    elif(emotion == "Trust"):
        urlhere='http://www.imdb.com/search/title?genres=western&title_type=feature&sort=
        moviemeter, asc'
```

```

# movie against emotion Surprise
elif(emotion == "Surprise"):
    urlhere='http://www.imdb.com/search/title?genres=film_noir&title_type=feature&sort=moviemeter, asc'

    response = HTTP.get(urlhere)
    data = response.text

    soup = SOUP(data, "lxml")

    title = soup.find_all("a", attrs = {"href" : re.compile(r'\title\/tt+\d*\')})
    return title

if __name__ == '__main__':
    emotion = input("Enter the emotion: ")
    a = main(emotion)
    count = 0

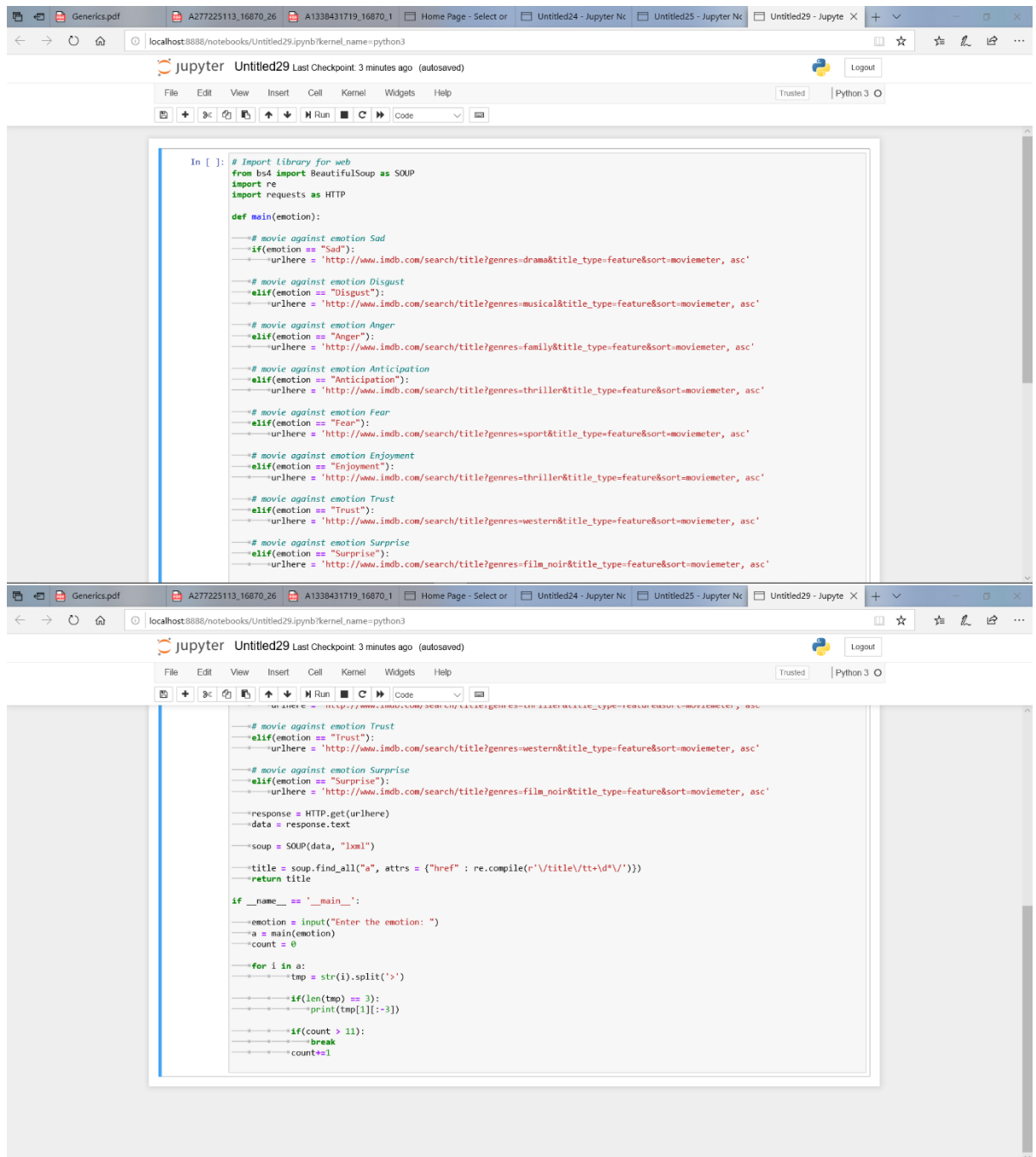
    for i in a:
        tmp = str(i).split('>')

        if(len(tmp) == 3):
            print(tmp[1][:-3])

    if(count > 11):
        break
    count+=1

```

Code Screenshots:



```
In [ ]: # Import library for web
from bs4 import BeautifulSoup as SOUP
import re
import requests as HTTP

def main(emotion):
    # movie against emotion Sad
    if(emotion == "Sad"):
        urlhere = 'http://www.imdb.com/search/title?genres=drama&title_type=feature&sort=moviemeter, asc'
    # movie against emotion Disgust
    elif(emotion == "Disgust"):
        urlhere = 'http://www.imdb.com/search/title?genres=musical&title_type=feature&sort=moviemeter, asc'
    # movie against emotion Anger
    elif(emotion == "Anger"):
        urlhere = 'http://www.imdb.com/search/title?genres=family&title_type=feature&sort=moviemeter, asc'
    # movie against emotion Anticipation
    elif(emotion == "Anticipation"):
        urlhere = 'http://www.imdb.com/search/title?genres=thriller&title_type=feature&sort=moviemeter, asc'
    # movie against emotion Fear
    elif(emotion == "Fear"):
        urlhere = 'http://www.imdb.com/search/title?genres=sport&title_type=feature&sort=moviemeter, asc'
    # movie against emotion Enjoyment
    elif(emotion == "Enjoyment"):
        urlhere = 'http://www.imdb.com/search/title?genres=thriller&title_type=feature&sort=moviemeter, asc'
    # movie against emotion Trust
    elif(emotion == "Trust"):
        urlhere = 'http://www.imdb.com/search/title?genres=western&title_type=feature&sort=moviemeter, asc'
    # movie against emotion Surprise
    elif(emotion == "Surprise"):
        urlhere = 'http://www.imdb.com/search/title?genres=film_noir&title_type=feature&sort=moviemeter, asc'

    response = HTTP.get(urlhere)
    data = response.text
    soup = SOUP(data, "lxml")
    title = soup.find_all("a", attrs = {"href" : re.compile(r'\/title\/tt+ld*\/')})
    return title

if __name__ == '__main__':
    emotion = input("Enter the emotion: ")
    a = main(emotion)
    count = 0

    for i in a:
        tmp = str(i).split('>')
        if(len(tmp) == 3):
            print(tmp[1][:3])
        if(count > 11):
            break
        count+=1
```

Output:

```
Enter the emotion: Sad
Once Upon a Time... in Hollywood
Contagion
Underwater
Knives Out
```

In []:

```
break
count+=1
```

```
Enter the emotion: Fear
The Way Back
Ford v. Ferrari
Fighting with My Family
The Big Lebowski
```

In []: