COSC 5369 Graduate Projects

Project Proposal

Project Title: Movie Recommendation System using a Hybrid Classifier

Supervisor: Dr. XXX

Committee Member:

Dr. XXX

Dr. XXX

Spring 2020

Note: Do not Copy and Paste any content from internet and published papers in your report. Write your report using your own language. Copy/Paste content in report will receive F for this course.

Format Requirement: (updated 1/22/2020)
Font: Times New Roman
Font Size: 10 pt
Line Spacing: 1.5 lines

Figure title has to be placed below each figure
Table title has to be placed above each table

Abstract

Every day a new multimedia streaming platform comes into the market but not all are successful. To be a successful multimedia streaming platform in the market providing the content for as low cost as possible isn’t the only factor to be considered. Providing the end user what to view or watch next is really important. The better the suggestions are the best the application is. This is what a recommender system does it suggests the content based on patterns of others users in system. The pattern observation can be done based on different attributes like ratings, watch time, categories, content etc., Apart from selection of attributes it is also important to select a right approach for the finding out the suggestions. These types of suggestions are always user specific. This proposal is to create a Movie Recommender System using a hybrid classifier (KMeans + Navie Bayes). Most of the time data will be unsupervised data. But supervised classifiers are always better and efficient, this approach will merge the unsupervised classifier with supervised classifier so that the final recommendations might be more efficient. The approach starts with collections of movies dataset and ratings dataset and once preprocessing is completed the data will be given KMeans to create the clusters and convert the data to supervised data. The supervised data is used to train the Navie Bayes model. This model is used to later to find out the patterns in the user watch history and provide the recommendations.

Keywords:

Table of Content

List of Figures

List of Tables

1. Introduction

The amount of data generated by the web is growing at an huge rate. The very fact is that more products and services are available to end users now than ever has, not only compounded the matter but has also increased its scope and diversity. This presents a major challenge to e-commerce because consumers cannot simply explore and compare every possible product. Businesses are forced to return up with smarter solutions to prevent consumers from being confused by seemingly countless choices and more importantly to be still relevant to them by directing their attention to commodities they might be curious about. Recommendation systems (RSs) are introduced to e-commerce to assist businesses tackle this challenge. Among all the prevailing techniques, collaborative filtering is one among the foremost promising approaches to build RSs. This task is accomplished by searching the database for user profiles with similar tastes.

Businesses that sell a humongous assortment of products on a day to day both in world like Walmart, Target, etc. on their e-business find RS immensely useful. RSs help businesses gauge consumer interests and retain them by actively engaging them. The system provides recommendations supported the ratings on commodities that users have previously rated. A RSs is efficient since in practice the system handles tens of thousands of ratings and prediction is calculated in real-time. The more data the RS has got to work with, more accurate the predictions done by the system are. Recommender systems are often classified supported the approach won't to provide recommendations and therefore the technique used.

The MovieLens [1] is a website which has these huge movies collection, along with the ratings collected from multiple users for each movie. Using these datasets RSs can build to recommend the movies. To improve the efficiency and increase the speed of the system a hybrid classifier is being used here. The dataset which has been collected will be processed so that it has the average rating of the individual movie along score i.e., numbers of users who has rated the movie and finally the average rating of each genre.

The processed dataset will be undergoing clustering process using KMeans [2] clustering algorithm. It uses recursion method to find n centroids for the given data. In this movie's dataset 100 clusters are required since the score of the movies will be in between from 0.1 to 10.0 every time. One the clustering is done the cluster value indicates the category of the movie. A movie might be on multiple clusters based on average genre rating and score.

The KMeans generated output is used to train the Navie Bayes [3] model. This model can be used to determine the patterns of any user's watch history and recommend movies to watch next. And one more thing to consider here is Navie Bayes only accept continuous data as input and provides output of nominal data.

$$\underset{\text{Posterior Probability}}{P(c \mid x)} = \frac{\overset{\text{Likelihood}}{P(x \mid c)}\,\overset{\text{Class Prior Probability}}{P(c)}}{\underset{\text{Predictor Prior Probability}}{P(x)}}$$

$$P(c \mid X) = P(x_1 \mid c) \times P(x_2 \mid c) \times \cdots \times P(x_n \mid c) \times P(c)$$

Figure 1.1 Navie Bayes Formula

A python webserver created using Flask [4] package will be providing a REST API which takes input of user's watch history and processes it using the trained Navie Bayes model to recommend the Movies. A webpage is used an interface to capture the user's input and display the output.

2. Literature Review

Read related papers (at least 5) and write one paragraph of summary of each paper. List all your referenced paper in the reference section using IEEE format as well.

3. Methodology

This proposed Movie Recommender System uses a Hybrid Classifier consisting of KMeans and Navie Bayes. KMeans is an unsupervised based clustering algorithm whereas Navie Bayes is an supervised based classification algorithm. The both combined to form a hybrid classifier working to find out the patterns in the users movie's watch history and recommend movies for them based on the patterns.

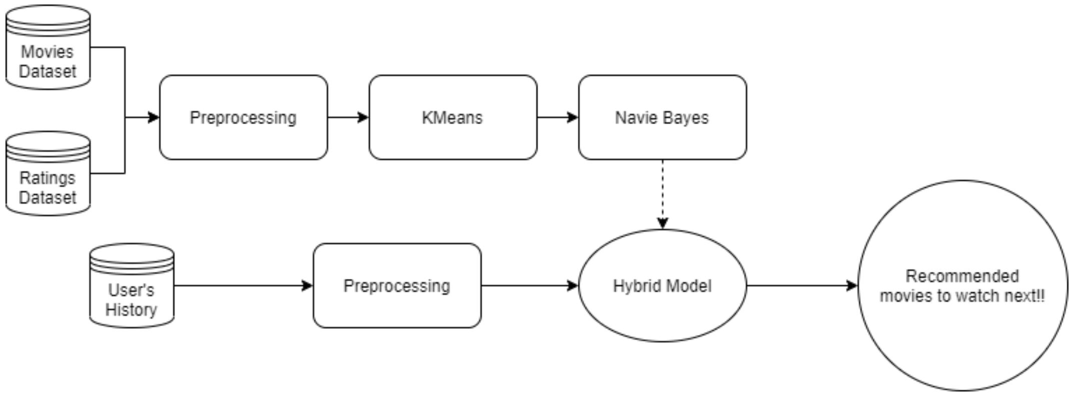

Figure 3.1 Architecture

The trained Navie Bayes model is used in the webserver to provide the movie recommendations in the real time to the users. The webserver consists of two endpoints. One of the endpoints is root which provides a web-based interface for user to provide input and view the output. The other endpoint is a REST API taking input as users watch history and processes it using Navie Bayes model and sends back the output.

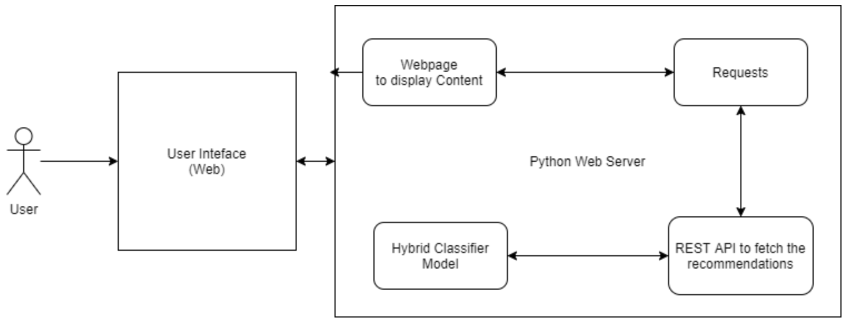

Figure 3.2 Working Process

3.1. Algorithms

a. KMeans

- Select the 'k' value i.e., the number of clusters you want to identify.

- Choose randomly ‘k’ data points as centroids (c1, c2…, ck) from the vector space.
- Repeat until convergence:
    - for each data point xi:
        - find the nearest centroid cj using Euclidian distance.
        - assign xi to that nearest cluster j.
    - for each cluster find the new centroid which is the mean of all the xi points assigned to that cluster j.
- Terminate when none of the cluster assignment change.

b. Navie Bayes

- Convert the data into frequencies
- Find the probabilities of the all possible outcomes using the frequencies
- Now using Navie Bayesian equation calculate the posterior probability

3.2. Environment Setup

3.2.1. Hardware Requirements

- CPU with minimum of 4 physical cores
- RAM with minimum of 8GB
- Hard Disk
- Operating System (Windows, Linux, Mac)

3.2.2. Software Requirements

- Python 3.7.x version
- Python packages required numpy, sklearn, pandas, jupyter, Flask
- Chrome Browser

3.3. Dataset

Dataset has been gathered from the MovieLens website. There are two files considered from entire sources.

3.3.1. movies.csv

This file consists of three fields movieId, title, genres.

movieId	title	genres
1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy
2	Jumanji (1995)	Adventure|Children|Fantasy
3	Grumpier Old Men (1995)	Comedy|Romance
4	Waiting to Exhale (1995)	Comedy|Drama|Romance
5	Father of the Bride Part II (1995)	Comedy
6	Heat (1995)	Action|Crime|Thriller

3.3.2. ratings.csv

This file consists of four fields userId, movieId, rating, timestamp.

userId	movieId	rating	timestamp
1	1	4	964982703
1	3	4	964981247
1	6	4	964982224

1	47	5	964983815
1	50	5	964982931
1	70	3	964982400

3.4. Code

```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import os
from itertools import chain
import matplotlib.pyplot as plt

input_dir = "./input/movielens-mllatestsmall"
chunk_size = 200000
movies_dataframe = pd.read_csv(input_dir + "/ml-latest-small/movies.csv",header=0)
movies_dataframe["genres"] = movies_dataframe["genres"].str.split("|")
ratings_dataframe = []
for chunk in pd.read_csv(input_dir + "/ml-latest-
small/ratings.csv",header=0,chunksize=chunk_size):
    chunk = chunk.drop(["timestamp"],axis=1)
    ratings_dataframe.append(chunk)
ratings_dataframe = pd.concat(ratings_dataframe,axis=0)
print("DONE READING")
print(movies_dataframe.shape)
print(movies_dataframe.head(1))
print("\n\n")
print(ratings_dataframe.shape)
print(ratings_dataframe.head(1))
print("\n\n")

ratings_dataframe = ratings_dataframe.groupby("movieId")
ratings_dataframe = pd.DataFrame({
    "movieId" : ratings_dataframe["movieId"],
    "score" : ratings_dataframe.count()["rating"],
    "rating" : ratings_dataframe.mean()["rating"]
})
ratings_dataframe = ratings_dataframe.drop("movieId",axis=1)
movies_dataframe = pd.merge(ratings_dataframe, movies_dataframe, on='movieId')
del ratings_dataframe

movies_dataframe

genres_lens = movies_dataframe['genres'].map(len)
movies_dataframe = pd.DataFrame({
    'movieId': np.repeat(movies_dataframe['movieId'], genres_lens),
    'score': np.repeat(movies_dataframe['score'], genres_lens),
    'rating': np.repeat(movies_dataframe['rating'], genres_lens),
    'title': np.repeat(movies_dataframe['title'], genres_lens),
    'genre': chain.from_iterable(movies_dataframe['genres'])
})
movies_dataframe.head(1)

genre_rating_avg = movies_dataframe.groupby("genre")["rating"].mean()
genre_rating_avg = pd.DataFrame({'genre':genre_rating_avg.index,
'genre_rating':genre_rating_avg.values})

movies_dataframe = pd.merge(movies_dataframe,genre_rating_avg,on="genre")

movies_dataframe

from sklearn.cluster import KMeans

kmeans =
KMeans(n_clusters=100).fit(movies_dataframe.drop(["movieId","genre","title"],axis=1))
centroids = kmeans.cluster_centers_
cluster_map = pd.DataFrame()
```

```python
cluster_map['category'] = kmeans.labels_
cluster_map
movies_dataframe = pd.merge(movies_dataframe,cluster_map,left_index=True,right_index=True)
movies_dataframe

fig = plt.figure(figsize=(20,10))
graph = fig.add_axes([0.1,0.1,0.8,0.8])
graph.scatter(movies_dataframe['movieId'], movies_dataframe['rating'], c= kmeans.labels_.astype(float), s=1, alpha=0.1)
graph.scatter(centroids[:, 0], centroids[:, 1], c='red', s=50)
plt.show()

fig = plt.figure(figsize=(20,10))
graph = fig.add_axes([0.1,0.1,0.8,0.8])
graph.scatter(movies_dataframe['movieId'], movies_dataframe['genre_rating'], c= kmeans.labels_.astype(float), s=1, alpha=0.1)
graph.scatter(centroids[:, 0], centroids[:, 1], c='red', s=50)
plt.show()


fig = plt.figure(figsize=(20,10))
graph = fig.add_axes([0.1,0.1,0.8,0.8])
graph.scatter(movies_dataframe['movieId'], movies_dataframe['score'], c= kmeans.labels_.astype(float), s=1, alpha=0.1)
graph.scatter(centroids[:, 0], centroids[:, 1], c='red', s=50)
plt.show()

movies_dataframe.to_csv("movies.csv", mode='w',index=False, header=True)
print("DONE WRITING")

from sklearn.naive_bayes import GaussianNB
model = GaussianNB()
model.fit(movies_dataframe.drop(["movieId","title","genre","category"],axis=1), movies_dataframe["category"])

test = pd.DataFrame({
    "movieId" : np.array([8644,32031,136800,140174,140627,141818,141994]),
    "title" : np.array(["I, Robot (2004)","Robots (2005)","Robot Overlords (2014)","Room (2015)","Battle For Sevastopol (2015)","Ordinary Miracle (1978)","Saving Christmas (2014)"]),
    "genres" : np.array(["Action|Adventure|Sci-Fi|Thriller","Adventure|Animation|Children|Comedy|Fantasy|Sci-Fi|IMAX","Action|Adventure|Sci-Fi","Drama","Drama|Romance|War","Comedy|Drama|Fantasy|Romance","Children|Comedy"])
})
test

test["genres"] = test["genres"].str.split("|")
genres_lens = test['genres'].map(len)
test = pd.DataFrame({
    'movieId': np.repeat(test['movieId'], genres_lens),
    'title': np.repeat(test['title'], genres_lens),
    'genre': chain.from_iterable(test['genres'])
})
most_frequent_genre = test.groupby("genre").count()["title"].max()
most_frequent_genre = test.groupby("genre").count().loc[test.groupby("genre").count()["title"] == most_frequent_genre]
most_frequent_genre

df = movies_dataframe.loc[movies_dataframe['genre'].isin(most_frequent_genre.index)]

genre_rating = df.groupby("genre")["rating"].mean()
genre_rating = pd.DataFrame({
    "genre" : genre_rating.index,
    "genre_rating" :  genre_rating.values
})

```

```
df = pd.merge(df.drop("genre_rating",axis=1),genre_rating,on="genre")
df["score"] = df["score"] + 1
category = model.predict(df.drop(["movieId","title","genre","category"],axis=1))
df["category"] = category

frequent_category = df.groupby("category").count()["title"]
frequent_category = frequent_category.loc[frequent_category.max() ==
    frequent_category.values].index

df = df.loc[df['category'].isin(frequent_category.values)]
df = df.loc[~df['movieId'].isin(test["movieId"])]
df = df.drop_duplicates(subset=['movieId',"title","score","rating"])

top100_watched = df.nlargest(100,["score"])
top10_rated = top100_watched.nlargest(10,["rating"])

top10_rated
```

4. Experimental Results

Discuss your experimental results in details here.

Include all tables and Figures/Screenshot of your experimental results

131. Conclusions and Suggestions for Future Week

132. Bibliography

1. MovieLens Datasets, https://grouplens.org/datasets/movielens/
2. K-Means Clustering, https://en.wikipedia.org/wiki/K-means_clustering
3. Navie Bayes Classification, https://en.wikipedia.org/wiki/Naive_Bayes_classifier
4. Python Flask, https://pypi.org/project/Flask/
5. Developers@Work, https://developerswork.online/

Appendix. User Guide and Reference

A. Include all information required for a future student or faculty member who wants to verify or extend your work. Such things as the parameters in a simulation, the necessary information in a configuration file, the source of software used in the Project but not written by the student, and syntax for accessing the programs should be part of the User Guide and Reference.
B. Well documented source code.