**A Project on**

A CLOUD BASED PERSONALIZED RECOMMENDER SYSTEM

**Master of Technology**

***In***

**Software Engineering**

***By***

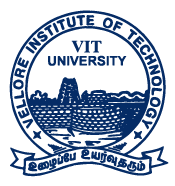
**P. RESHMA 17MIS1009**

**P. NITYA SREE 17MIS1007**

**N. NIMISHA YADAV 17MIS1183**

***Under the guidance of***

**Prof Muthumanikandan**

**

**Department of Computing Science And Engineering**

**Vellore Institute of Technology**

**VIT University , Chennai Campus**

**Chennai ,India.June,2020.**



**Department of Computing Science and Engineering**

**Vellore Institute of Technology**

**Declaration**

We, hereby declare that the project entitled “**A Cloud Based Personalized Recommender System**” which is submitted by us to department of Computing Science and Engineering, Vellore Institute of Technology, VIT University Chennai, in partial fulfillment of the requirement for the award of the degree of Master of Technology in Software Engineering, has not been previously formed the basis for the award of any degree, diploma or other similar title or recognition.

P. Reshma

P. Nitya Sree

N. Nimisha Yadav

**Signature**

Prof. Muthumanikandan V

Assistant Professor

SCOPE

**Certificate**

This is to certify that the report “**A Cloud Based Personalized Recommender System”** is prepared and submitted by P. Reshma (17MIS1009), P. Nitya Sree (17MIS100), N. Nimisha Yadav (17MIS1183) to VIT Chennai, in partial fulfillment of the requirement for the award of the degree of Master of Technology in Software Engineering (5 year Integrated Programme) is a bona-fide record carried out under my guidance. The project fulfills the requirements as per the regulations of VIT and in our opinion meets the necessary standards for submission. The contents of this report have not been submitted and will not be submitted either in part or in full, for the award of any other degree or diploma and the same is certified.

Guide

Prof Muthumanikandan V

02-06-2020

**Acknowledgement**

We obliged to give our appreciation to a number of people without whom we could not have completed this thesis successfully.

We would like to place on record my deep sense of gratitude and thanks to my project guide Prof. Muthumanikandan V, School of Computer Science and Engineering (SCOPE), Vellore Institute of Technology, Chennai, whose esteemed support and immense guidance encouraged me to complete the project successfully.

Special mention to our Dean, Associate Dean, School of Computer Science and Engineering (SCOPE), Vellore Institute of Technology, Chennai, for motivating us in every aspect of software engineering.

We thank our management of Vellore Institute of Technology, Chennai, for permitting me to use the library and laboratory resources. We also thank all the faculty members for giving us the courage and the strength that we needed to complete our goal. This acknowledgment would be incomplete without expressing the whole hearted thanks to our family and friends who motivated us during the course of our work.

- P. Reshma ,P. Nitya Sree, N. Nimisha Yadav

|  |  |  |
| --- | --- | --- |
|  | iv | |
| 1 | 1 | |
|  | 1 | |
|  |  | |
|  |  | |
|  |  | |
|  | |
|  |  | |
|  |  | |
|  |  | |
|  |  | |
|  |  | |
|  |  | |
|  |  | |
|  |  | |
| 3 |  | |
|  |  | |
|  |  | |
|  |  | |
| 4 |  | |
|  |  | |
|  |  | |
|  |  | |
| 5 |  | |
|  |  | |
| 6 |  | |
|  |  | |
|  |  | |
|  |  | |

Abstract

1. Introduction

1.1 Background . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

1.2 Statement . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

1.3 Motivation . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

1.4 Challenges . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

2. Planning & Requirements Specification

2.1 Literature Review……………………………….

2.2 System Planning . . . . . . . . . . . . . . . . . . . . . . . . . .

2.3 Requirements . . . . . . . . . . . . . . . . . . . . . . . . . . . .

2.3.1 User requirements . . . . . . . . . . . . . . . . . . . . .

2.3.2 Non-Functional requirements . . . . . . . . . . . . . . .

2.4 System Requirements . . . . . . . . . . . . . . . . . . . . . . . .

2.4.1 Hardware Requirements . . . . . . . . . . . . . . . . . .

2.4.2 Software Requirements . . . . . . . . . . . . . . . . . . .

3. System Design

4. Implementation of System /Methodology

4.1 Content Based Technique…………………………….

4.2 Model Based Collaborative………………………………

4.2.1 Clustering CF…………………………………………

4.2.2 Matrix Factorization……………………………..

4.3 Memory based collaborative……………………………..

4.4 Deep Learning…………………………………………..

5. Results & Discussion

6. Conclusion and Future Work

References

Appendix <Sample code, snapshot etc.>

**Abstract:**

Now-a-days, many major e-commerce Websites are using recommendation systems to provide relevantsuggestions to their customers. The recommendations could be based on various parameters, such as items popular on the company’s Website; user characteristics such as geographical location or other demographic information; or past buying behaviour of top customers. In this paper, a book recommendation engine is proposed which uses data mining techniques for recommending books, movies, songs etc The proposed recommender system will give its users‟ the ability to view and search books as well as novels which will be used to draw out conclusions about the stream of a user and the genre of the books liked by that user. The system will analyze the user behaviour by using the features of various recommendation techniques such as content based; collaborative and demographic. Thus, in this paper a hybrid recommendation system is proposed which satisfies a user by providing best and efficient recommendations.

**Keywords:** Model-based Collaborative technique, memory-based Collaborative technique, Content Based Technique, Recommendation Engine,User’s Interest, deep learning, matrix factorization

**1 Introduction**

* 1. *Background:*

With the ever-growing volume of online information, recommender systems have been an effective strategy to overcome such information overload. Utility of recommender systems cannot be overstated, given its widespread adoption in many web applications, along with its potential impact to ameliorate many problems related to over-choice. In recent years, recommendation has garnered considerable interest in many research owing not only to stellar performance but also the are active property of learning feature representations from scratch. so we decided to build a recommendation system

* 1. *Statement:*

Build recommendation system to give user recommendation about books, movies in a single portal

*1.3 Motivation:*

The main focus of this thesis is two fold:

•build a recommender system,

•find the best algorithm

* 1. *Challenges:*

Lack of Data. Perhaps the biggest issue facing **recommender systems** is that they need a lot of data to effectively make **recommendations**.

 Changing Data.

 Changing User Preferences.

 Unpredictable Items.

**2. Planning and Requirements Specification**

*2.1 Literature Review:*

Itinerary recommendation systems Madhu & Manjula (2016a) proposed an automatic location recommendation system, enhanced in this work with the addition of itinerary recommendation. Choudhury et al. (2010) discussed the automatic generation of itineraries from a POI graph created from photo streams taken by users. However, it offers only an approximate solution, coupled with a longer execution time and requiring more user intervention

Dunstall et al. (2004) proposed an automated itinerary planning system for holiday travel, a purely commercial model requiring greater user intervention and requiring more execution time.

Activity/event recommendation systems Zheng et al. (2010) proposed a recommendation system for recommending locations and activities. The system uses a CF-based technique for recommendation and maintains three matrices: location-activity, locationfeature and activity-activity matrix. When a user logs in for an activity, data maintained in the location-activity matrix provides information to the effect that the user has just been associated with an activity. The location-activity matrix displays the association between locations and categories of points of interest. Similar locations will have similar, possible activities.

Zheng et al. (2009) proposed a user recommendation system which identifies expert users by deploying the HITS algorithm. The algorithm is applied over a hierarchical graph, built using users historical trajectories. Friends can also be recommended to users, based on this method, by 20 following the links. The node or person connected with the most number of links will be the expert or celebrity, depending on the context

User recommendation systems Ying et al. (2010) proposed a friend recommendation system by following a systematic approach. Users travel routes are converted to a sequence of locations and a mining algorithm used to discover patterns in the routes. Similarities between patterns are identified and friends recommended, based on the similarities identified.

Social media recommendation systems A social media recommendation system recommends media in social networks or internet like online news, Twitter pages, online videos etc. to users. Sandholm & Ung (2011) proposed a social media recommendation system for online web content. It is built on a CF-based method which considers geographical influences on ratings.

*2.2 System plan:*

The input to our system are API requests, which can be classified as online or batch:Online requests, which must be handled in real time. Their processing cant be delayed, because users are waiting for a response. They are also utilized to update the profile for new users and begin to provide them with recommendations. The request processor evaluates if a certain API request needs to be run online or it can be batched.Batch requests, which may be stored and processed only at given time periods.

Now requests processing can be delayed and attended when the system is not at full capacity. These requests are used to upload the initial data from a client and also to update information concerning users with a wide user profile.. Each API request generates an HTTP request to a certain end-point where the Request Processor evaluates it and deter-mines whether it must be processed at that moment or it can be delayed until more requests reach the system (for amore optimum processing) or until certain batch process is programmed to be run. Requests can also be classified as update or retrieval:

Retrieval requests just ask the system to return some kind of information, such as a recommendation.Update requests have the objective to update the pro-file of the source user. When an update request begins to be processed there are two steps that must be taken to produce recommendations for the user.

As we wanted to be able to process several types of recommendations (collaborative filtering, content based, social recommendations), the system had to be general enough to process data in several ways. So we defined those steps in a way that enabled the use of any possible recommender algorithm Update user profile.

This can be done by recalculating similarity with other users, re-calculating trust or updating a content-based profile. Update user recommendations.

This step uses the values obtained from the previous task as input for there commendation algorithm and produces a new rank of recommendations for the user

*2.3 Requirements*

# 2.3.1 User requirements

## 1. Collect and organize information on users and products

This is the essential first step. You need to know who your users are and what they are using. In our case, it was Klips, the data visualizations that drive engagement with data that Klipfolio users connect to in the product.

## 2. Compare User A to all other users

Using those standard forms, you next design a function that compares User A to all other users.

This function should create a set of users (along with the Klips that each has used) that are most similar to User A.

Using common machine learning libraries like [Python's scikit-learn](http://scikit-learn.org/stable/), we are able to use the Nearest Neighbours algorithm out of the box on our transformed data to compute this user set.

## 3. Create a function that finds products that User A has not used, but which similar users have

Since we discovered that Lianne, Luke, and Alex are most similar to User A, we can examine each users vector to determine the Klips which are new to User A but used by these similar neighbours.

This can be done using basic set theory operations on the set of Klips used by the neighbours, and the set of Klips used by User

## 4. Rank and recommend

If we want to interest User A in new products, we will increase our chances of success by assigning a higher rank to products that customers similar to User A already use.

We can extend the recommendation system by ranking the recommended items to User A. The greater the number of similar customers using a Klip, the higher the rank that Klip gets assigned.

# 2.3.2 Non functional requirements

# The requirements in this section provide a detailed specification of the user interaction with the software and measurements placed on the system performance.

# Design constraints

# This section includes the design constraints on the software caused by the hardware.

# Hard Drive Space:

# The applications need of hard drive space of No more than 20 MB.

# Maintainability:

The application should be easy to extend. The code should be written in a way that it favours implementation of the functions.

System Reliability:

The reliability that the system gives the right result on a search.

Availability:

System Availability: The availability of the system when it is used. The average system availability (not considering network failing).

*2.4 System Requirements*

**2.4.1 Hardware Requirements**

Computer/mobile

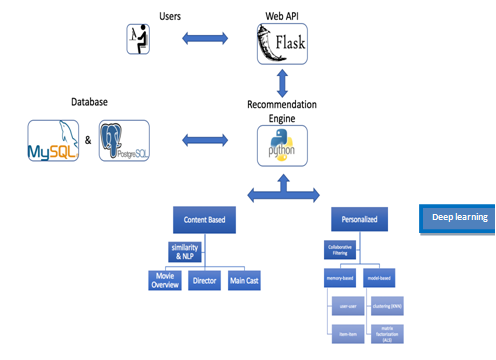
Min 2gb ram

Min 20 mb memory space

**2.4.2 Software Requirements**

Min Windows xp

**3. System Design**



In above figure , the architecture of proposed system is shown. The main module in this system is Recommender system. The registered user logs in to the system.

The user can view books,movies of different categories. The user can also rate books as per his/her likings. The rating and searching history of books, movies for each individual is stored in the database. In recommender system module, mainly three techniques are used for recommendations.

Collaborative based filtering and content based filtering techniques are performed on the data which is present in user’s history. If null results are generated from these techniques then demographic recommender is used. The results from all the recommender techniques are combined and the set for recommended books is generated.

**4. Implementation of System/Methodology**

**Techniques used**

Recommendation techniques have a number of possible classifications. The classification is based on the sources of data on which recommendation is based and the use to which that data is put. In general, recommender systems have

1. background data, the information that the system has before the recommendation process begins,
2. input data, the information that user must communicate to the system in order to generate a recommendation, and
3. an algorithm that combines background and input data to arrive at its suggestions.
4. The recommendation techniques are classified into five types: 1]model based Collaborative. 2] Content based. 3] memory based collaborative. 4] deep learning

**4.1 Content Based Technique**

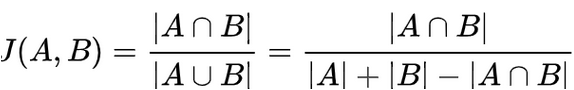
This approach relies on creating a plethora of parameters to describe a product „P‟. Considering a smart phone as an example the possible parameters could be screen size, image quality, Wi-Fi protocols, brand names, operating systems etc.

The larger the parameter set the better and easier it is to match patterns with user profile and his online footprint. The parameters can then be assigned weights and hence a relative priority is set for each of the parameter. All these parameters are then used to create a user profile and each time a prospective user checks out another product, his profile gets updated.

Hence we see that the system learns about the user preferences and selection patters by his online footprint. Popular platforms that use such an approach are IMDB and Pandora.For Content based technique, Locality-sensitive hashing method is used.Locality-sensitive hashing (LSH) is a method of performing probabilistic dimension reduction of high-dimensional data.

The basic idea is to hash the input items so that similar items are mapped to the same buckets with high probability. In LSH the goal is to maximize probability of "collision" of similar items. Jaccard similarity is used along with LSH method.

**The math**

****

**4.2 Model Based Collaborative**

Model-based CF Complex patters which are based on training data, are the models (such as data mining algorithms, machine learning) and then intelligent predictions are made for CF tasks for the real world data which are based on learnt models.

It intuitive rationale for recommendations. Model disadvantage of model-based CF is that it loses useful information for dimensionality reduction techniques .Model-based collaborative filtering :The main drawback of memory-based technique is the requirement of loading a large amount of in-line memory.

The problem is serious when rating matrix becomes so huge in situation that there are extremely many persons using system. Computational resource is consumed much and system performance goes down; so system cant respond user request immediately.

Model-based approach intends to solve such problems. There are four common approaches for model-based CF such as clustering, classification, latent model, Markov decision process (MDP), and matrix factorization.

**4.2.1. Clustering CF :** Clustering CF is based on assumption that users in the same group have the same interest; so they rate items similarly. Therefore users are partitioned into groups called clusters which is defined as a set of similar users. Suppose each user is represented as rating vector denoted ui = (ri1, ri2,, rin). The dissimilarity measure between two users is the distance between them.

We can use Minkowski distance, Euclidian distance or Manhattan distance.The less distance(u1, u2) is, the more similar u1 and u2 are. Clustering CF includes two steps: 1. Partitioning users into clusters and each cluster always contains rating values. For example, every cluster resulted from k-mean algorithm has a mean which is a rating vector like user vector. 2. The concerned user who needs to be recommended is assigned to concrete cluster and her/his ratings are the same to ratings of such cluster. Of course how to assign a user to right cluster is based on the distance between user and cluster.

So the most important step is how to partition users into clusters. There are many clustering techniques such as k-mean and k-centroid. The most popular clustering algorithm is k-mean algorithm [3] which includes three following steps:

1. It randomly selects k users, each of which initially represents a cluster mean. Of course, we have k cluster means. Each mean is considered as the representative of one cluster. There are k clusters.

2. For each user, the distance between it and k cluster means are computed. Such user belongs to the cluster to which it is nearest. In other words, if user ui belong to cluster cv, the distance between ui and mean mv of cluster cv, denoted distance(ui, mv), is minimal over all clusters.

3. After that, the means of all clusters are re-computed. If stopping condition is met then algorithm is terminated, otherwise returning step 2. This process is repeated until the stopping condition is met. There are two typical terminating conditions (stopping conditions) for k-mean algorithm: - The k means are not changed. In other words, k clusters are not changed. This condition indicates a perfect clustering task. - Alternatively, error criterion is less than a pre-defined threshold.

**4.2.2 Matrix Factorization:**

In the previous attempt, I have used memory-based collaborative filtering to make movie recommendations from users ratings data. I can only try them on a very small data sample (20,000 ratings), and ended up getting pretty high Root Mean Squared Error (bad recommendations).

Memory-based collaborative filtering approaches that compute distance relationships between items or users have these two major issues: It doesn’t scale particularly well to massive datasets, especially for real-time recommendations based on user behavior similarities which takes a lot of computations. Ratings matrices may be overfitting to noisy representations of user tastes and preferences. When we use distance based neighbourhood approaches on raw data, we match to sparse low-level details that we assume represent the users preference vector instead of the vector itself.

Thus we need to apply Dimensionality Reduction technique to derive the tastes and preferences from the raw data, otherwise known as doing low-rank matrix factorization.

Why reduce dimensions? I can discover hidden correlations / features in the raw data. I can remove redundant and noisy features that are not useful. I can interpret and visualize the data easier.I can also access easier data storage and processing.

The Math

Model-based Collaborative Filtering is based on matrix factorization (MF) which has received greater exposure, mainly as an unsupervised learning method for latent variable decomposition and dimensionality reduction.

Matrix factorization is widely used for recommender systems where it can deal better with scalability and sparsity than Memory-based CF: The goal of MF is to learn the latent preferences of users and the latent attributes of items from known ratings (learn features that describe the characteristics of ratings) to then predict the unknown ratings through the dot product of the latent features of users and items.

When you have a very sparse matrix, with a lot of dimensions, by doing matrix factorization, you can restructure the user-item matrix into low-rank structure, and you can represent the matrix by the multiplication of two low-rank matrices, where the rows contain the latent vector.

You fit this matrix to approximate your original matrix, as closely as possible, by multiplying the low-rank matrices together, which fills in the entries missing in the original matrix. A well-known matrix factorization method is Singular value decomposition (SVD).

At a high level, SVD is an algorithm that decomposes a matrix A into the best lower rank (i.e. smaller/simpler) approximation of the original matrix A. Mathematically, it decomposes A into a two unitary matrices and a diagonal matrix: where A is the input data matrix (user’s ratings), U is the left singular vectors (user features matrix), Sum is the diagonal matrix of singular values (essentially weights/strengths of each concept), and V^T is the right singular vectors (movie features matrix).

U and V^T are column orthonormal, and represent different things: U represents how much users like each feature and V^T represents how relevant each feature is to each movie. To get the lower rank approximation, I take these matrices and keep only the top k features, which can be thought of as the underlying tastes and preferences vectors.

The Evaluation for loss function

****

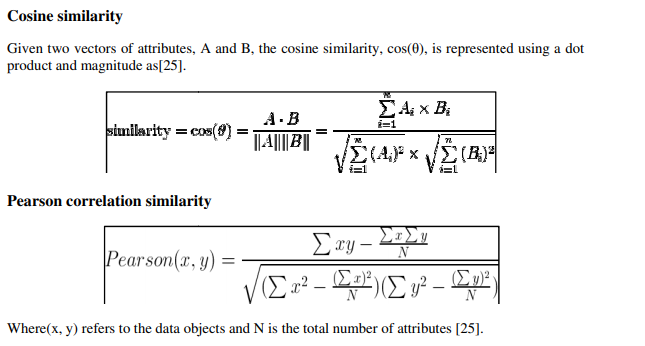
**4.3 Memory based collaborative:**

A memory-based CF (nearest-neighbor) approach, mostly called as a form of implementation of the Word of Mouth phenomenon (Jin, Chai & Si, 2004) since the entire user database with their preferences are kept in memory.

For each prediction computation is performed K. Madadipouya, S. Chelliah - A Literature Review on Recommender Systems Algorithms, Techniques and Evaluations 113 on the whole database. This method could predict a user interests on a specific item based on the rating information of similar user profiles. It reflects where the prediction of a specific item (belonging to a specific user) is done by sorting the row vectors (user profiles) by its dissimilarity toward the user.

In this method, more rating by more similar users leads to more rating prediction. Various types of memory-based recommender systems have been developed. Decker and Lenz (2007) stated that Goldberg on 1992 developed certain type of memory-based CF system which is called Tapestry.

This approach mostly is used in information retrieval systems. Apart from developments which have been done by researchers, some commercial websites also have developed their own version of memory-based collaborative filtering

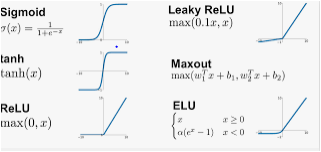


**4.4 Deep Learning**

The idea of using deep learning is similar to that of Model-Based Matrix Factorization. In matrix factorization, we decompose our original sparse matrix into product of 2 low rank orthogonal matrices. For deep learning implementation, we don’t need them to be orthogonal, we want our model to learn the values of embedding matrix itself.

The user latent features and movie latent features are looked up from the embedding matrices for specific movie-user combination. These are the input values for further linear and non-linear layers. We can pass this input to multiple relu, linear or sigmoid layers and learn the corresponding weights by any optimization algorithm (Adam, SGD, etc.).

This model performed better than all the approaches I attempted before (content-based, user-item similarity collaborative filtering, SVD). I can certainly improve this models performance by making it deeper with more linear and non-linear layers.

****

**5. Results and discussion:**

We first started off with context based model, then proceeeded with model and memory based collaborative method finally we performed deep learning method. The accuracy of deep learning method was the highest.

**6. Conclusion and future work:**

Recommender systems are an extremely potent tool utilized to assist the selection process easier for users. The implemented recommendation engine is a competent system to recommend Books for e-users. This recommender system will definitely be a great web application implemented in Java language.

Such type of web application will be proved beneficial for today‟s high demanding online purchasing web sites. This hybrid recommender system is more accurate and efficient as it combines the features of various recommendation techniques.The recommendation engine will reduce the overhead associated with making the best choices of books among the plenty.The future work can be focussed on improving the speed of the algorithm

**7. References**

[1] G. Adomavicius and A. Tuzhilin, Toward the next generation of recommender systems: A survey of the state-of the-art and possible extensions, IEEE Trans. Knowl. Data Eng.

[2] G. Linden, B. Smith, and J. York, Amazon recommendations: Itemto-item collaborative filtering, IEEE Internet Comput., Feb. 2003.

[3] Michael Hashler, Recommender Lab: A Framework for Developing and Testing Recommendation Algorithms Nov. 2011.

[4]R. Bell, Y. Koren, and C. Volinsky, Modeling relationships at multiple scales to improve accuracy of large recommender systems KDD '07: Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining, New York, NY, USA, 2007, ACM.

[5] O. Celma and P. Herrera, A new approach to evaluating novel recommendations, RecSys '08: Proceedings of the 2008 ACM conference on Recommender systems, New York, NY, USA, 2008, ACM.

[6] C. N. Ziegler, S. M. McNee, J. A. Konstan, and G. Lausen, Improving recommendation lists through topic diversification: Proceedings of the 14th international conference on World Wide Web, New York, USA, 2005, ACM.

[7] Robin Burke, Hybrid Recommender Systems: Survey and Experiments, California State University, FullertonDepartment of Information Systems and Decision Science

**Apendix**

**Sample code:**

Loading dataset

# Import libraries

%matplotlib inline

import math

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

# Reading ratings file

ratings = pd.read\_csv('ratings.csv', sep='\t', encoding='latin-1',

usecols=['user\_id', 'movie\_id', 'user\_emb\_id', 'movie\_emb\_id', 'rating'])

max\_userid = ratings['user\_id'].drop\_duplicates().max()

max\_movieid = ratings['movie\_id'].drop\_duplicates().max()

# Reading ratings file

users = pd.read\_csv('users.csv', sep='\t', encoding='latin-1',

usecols=['user\_id', 'gender', 'zipcode', 'age\_desc', 'occ\_desc'])

# Reading ratings file

movies = pd.read\_csv('movies.csv', sep='\t', encoding='latin-1',

usecols=['movie\_id', 'title', 'genres'])

Code for google drive:

from pydrive.auth import GoogleAuth

gauth = GoogleAuth()

gauth.LocalWebserverAuth() # Creates local webserver and auto handles authentication.

from pydrive.drive import GoogleDrive

# Create GoogleDrive instance with authenticated GoogleAuth instance.

drive = GoogleDrive(gauth)

# Create GoogleDriveFile instance with title 'Hello.txt'.

file1 = drive.CreateFile({'title': 'Hello.txt'})

filepath ="C:/Users/R078tu/Documents/web client 1/movpred.txt"

file1.SetContentFile(filepath)

file1.Upload() # Upload the file.

