

ITM UNIVERSITY GWALIOR

Jhansi Rd, Turari, Gwalior, Madhya Pradesh, 474001

School of Engineering & Technology

Department of Computer Science and Application

Batch 2020-24

CSD0603[P]: Minor Project

Submitted By :-

* Bhawesh Agrawal (BETN1CS20101)
* Prabhanshu Sisodia (BETN1CS20030)
* Akul Gupta (BETN1CS20003)

**Under the Supervision of :-**

Mr. Sheo Kumar, Assistant Professor

Department of CSA ITM University

Index

S. No. Topic

|  |  |
| --- | --- |
| I | Abstract |
| II | Objectives |
| III | Introduction |
| IV | Requirement of project (h/w, s/a, etc.) |
| V | Project Module Collaborative Filtering using Item to Item Similarity |
| VI | Overview of Python Flask Framework |
| VII | Project Screenshots |

1. Abstract

Recommender systems are an important part of the information and e-commerce ecosystem. They represent a powerful method for enabling users to filter through large information and product spaces. Nearly two decades of research on collaborative filtering have led to a varied set of algorithms and a rich collection of tools for evaluating their performance.

Research in the field is moving in the direction of a richer understanding of how recommender technology may be embedded in specific domains. The differing personalities exhibited by different recommender algorithms show that recommendation is not a onesize-fits-all problem. Specific tasks, information needs, and item domains represent unique problems for recommenders, and design and evaluation of recommenders needs to be done based on the user tasks to be supported.

Effective deployments must begin with careful analysis of prospective users and their goals. Based on this analysis, system designers have a host of options for the choice of algorithm and for its embedding in the surrounding user experience. This paper discusses a wide variety of the choices available and their implications, aiming to provide both practitioners and researchers with an introduction to the important issues underlying recommenders and current best practices for addressing these issues.

1. Objectives

Overall, the project aims to provide a good platform for designers to evaluate recommender systems and guide them to design better recommender systems. Based on the project’s specification, there are main two objectives extracted. Blow is the brief description of each projective.

1. A web application. A web application is developed to provide GUI interfaces for users to conduct experiments more easily and conveniently. The implementation process goes from prototyping, to the design of pages, to coding, and finally to optimisation. Along the way, the key idea that has been bearing in mind is making the front interfaces as easily interactive as possible.

1. Implementation of three algorithms. The three algorithms are all collaborative filtering ones, user-based, item-based and matrix factorization.

III. Introduction

What Is a Recommendation System?

A recommendation system is an artificial intelligence or AI algorithm, usually associated with machine learning, that uses Big Data to suggest or recommend additional products to consumers. These can be based on various criteria, including past purchases, search history, demographic information, and other factors. Recommender systems are highly useful as they help users discover products and services they might otherwise have not found on their own.

Recommender systems are trained to understand the preferences, previous decisions, and characteristics of people and products using data gathered about their interactions. These include impressions, clicks, likes, and purchases. Because of their capability to predict consumer interests and desires on a highly personalized level, recommender systems are a favorite with content and product providers. They can drive consumers to just about any product or service that interests them, from books to videos to health classes to clothing.

What is e-commerce?

E-commerce (electronic commerce) is the buying and selling of goods and services, or the transmitting of funds or data, over an electronic network, primarily the internet. These business transactions occur either as businessto-business (B2B), business-to-consumer (B2C), consumer-to-consumer or consumer-to-business.

The terms e-commerce and e-business are often used interchangeably. The

term e-tail is also sometimes used in reference to the transactional processes that make up online retail shopping.

1. Requirement of project

* + A desktop/laptop

* + Python 3 or above in system

* + Internet Connection with at least 64Kbps bandwidth

* + More than 8GB of RAM and at least i3 core processor

* + Required libraries:-

* + 1. Flask
    2. Numpy
    3. Pandas
    4. Sklearn
    5. Json
    6. Pickle

1. Project Module Collaborative Filtering using Item to Item Similarity

Collaborative Filtering is a technique or a method to predict a user’s taste and find the items that a user might prefer on the basis of information collected from various other users having similar tastes or preferences. It takes into consideration the basic fact that if person X and person Y have a certain reaction for some items then they might have the same opinion for other items too.

The two most popular forms of collaborative filtering are:

* User Based: Here, we look for the users who have rated various items in the same way and then find the rating of the missing item with the help of these users.
* Item Based: Here, we explore the relationship between the pair of items (the user who bought Y, also bought Z). We find the missing rating with the help of the ratings given to the other items by the user.

Let’s talk about Item-Based Collaborative Filtering in detail. It was first invented and used by Amazon in 1998. Rather than matching the user to similar customers, item-to-item collaborative filtering matches each of the user’s purchased and rated items to similar items, then combines those similar items into a recommendation list. Now, let us discuss how it works.

Item to Item Similarity: The very first step is to build the model by finding similarity between all the item pairs. The similarity between item pairs can be found in different ways. One of the most common methods is to use cosine similarity.

Formula for Cosine Similarity:



Prediction Computation: The second stage involves executing a recommendation system. It uses the items (already rated by the user) that are most similar to the missing item to generate rating. We hence try to generate predictions based on the ratings of similar products. We compute this using a formula which computes rating for a particular item using weighted sum of the ratings of the other similar products.

VI. Overview of Python Flask Framework

Web apps are developed to generate content based on retrieved data that changes based on a user’s interaction with the site. The server is responsible for querying, retrieving, and updating data. This makes web applications to be slower and more complicated to deploy than static websites for simple applications. There are two primary coding environments for the whole web app ecosystem. This article will give an overview of the Python Flask Framework and Its best practices.

Client-side Scripting

The code executed on the user’s browser visible to anyone who has access to the system, generating the first results.

Server-side Scripting

This type of code is run on the backend on a web server. To enable developers to design, build, maintain, host web apps over the internet, a web framework necessary.

A web framework is an architecture containing tools, libraries, and functionalities suitable to build and maintain massive web projects using a fast and efficient approach. They are designed to streamline programs and promote code reuse.

Python is home to numerous such frameworks, famous among which are Django and Flask. Its framework is a lightweight micro-framework based on Werkzeug, Jinja2.

The layout of the Python Flask Framework

* Module Init - (project\_root/app\_name/admin/\_\_init\_\_.py) - required to enable the app
* Module URL - (project\_root/app\_name/admin/url.py) - Url definitions of each module
* App root Init - (project\_root/app\_name/\_\_init\_\_.py) - Not necessary to define the entire app within \_\_init\_\_.py
* Module Views - (project\_root/app\_ame/admin/views.py) - Defines views for each module. Separate ‘.py.’ Files as the project scale to ensure they are accessible to URLs.
* Module Templates -

(project\_root/app\_name/admin/templates/admin/main.html) - Normal template folder.

HTTP Methods

Request

To process incoming data in Flask, you need to use the request object, including mime-type, IP address, and data. HEAD: Un-encrypted data sent to server w/o response.

GET

Sends data to the server requesting a response body.

POST

Read form inputs and register a user, send HTML data to the server are methods handled by the route. Flask attaches methods to each route so that different view functions can handle different request methods to the same URL.

Response

Flask invokes a view function. It has to return a response value to the client. HTTP requires it to be more than a string response, a status code.

* Informational – 1xx
* Successful – 2xx
* Redirection – 3xx Client Error – 4xx
* Server Error – 5xx

Templates

To maintain the site. Flask uses a powerful template engine, ‘Jinja2’, in its simplest form. A Jinja2 template is a file that contains the text of a response, returned by a view function that has a dynamic component represented by a variable.

Linking

Dynamic url routing support is included using ‘url\_for()’ helper function. For example, url\_for('sagar', name='project\_file', \_external=True) would return http://localhost:5000/sagar/project\_file.

Security

CSRF(Cross-Site-Request-Forgery) occurs when a malicious website sends requests to a different website on which the victim logs in. Flask-WTF protects against all such attacks. Apart from that, Flask also implements some common security mechanisms like session-based management, role mgmt, password hashing, basic HTTP and token-based authentication, optional log-in tracking.

Database Connectivity

Flask has no restrictions for the use of databases; there’s no native support for databases. However, they can be broadly divided into two categories.

* That following relational model for, e.g., SQL, sqlite3 mainly for structured data.
* That not following the relational model for, e.g., NoSQL primarily for unstructured data.

Flask-SQLAlchemy is a Flask extension that simplifies the use of

SQLAlchemy inside Flask applications. SQLAlchemy is a robust relational database framework that supports several databases back ends. It offers a high-level ORM and low-level access to the database’s native SQL functionality.

VII. Project Screenshots



















