3030&7130ICT
Data Analytics

Lab 9 – Data Analytics for Personalized Data

Table of Contents

1. Objectives

You will build a powerful movie recommender 🎬 and deploy it in a Flask application.

Instructions

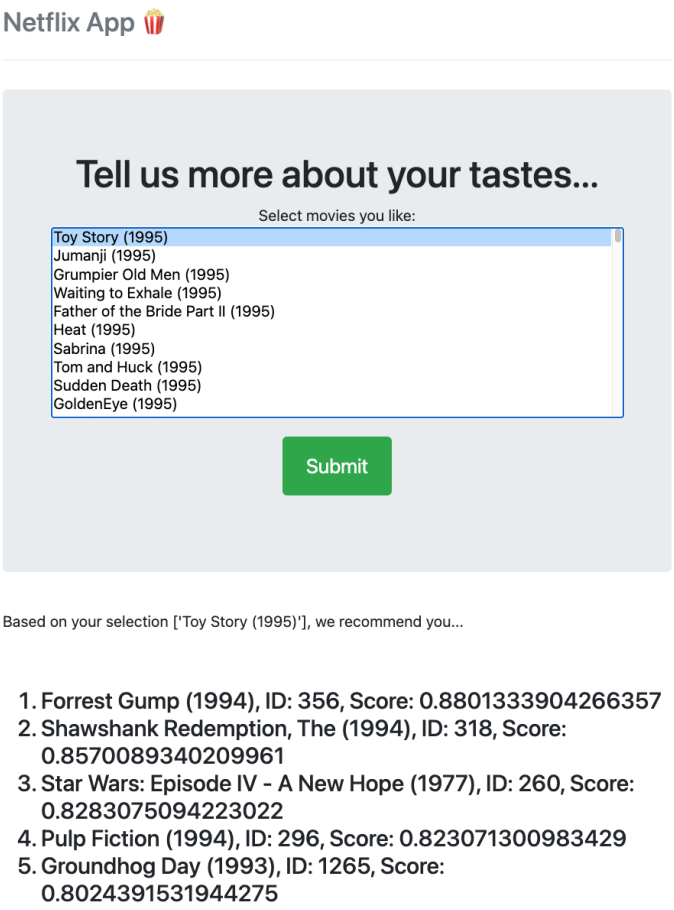
This algorithm will work like this:

- You (or any user) selects a few movies he/she likes
- The model returns a list of N movies recommendations corresponding to user's tastes

Before starting your project, please review the tutorial for Recommendation

This project has three steps :

1. First you will **prepare the data** so that we can successfully train a LightFM model;
2. Then you will **train the model**, and **test your recommendations;**
3. Optionally you will deploy your model into a small web-app, to present your recommendations in a more ergonomic way using Flask



LightFM installation

We will use the open-source library LightFM which provides easy python implementation of **hybrid** recommendation engines.

There are many other libraries in Python for building recommendation engines (ex: Crab, Surprise, etc.). We choose LightFM because it is easy to implement and very powerful (especially if you build hydrid engines).

As an example, LightFM was largely used by researchers in RecSys 2018 (challenge held by Spotify during one of the most popular conference in the field of recommender systems).

⚙ You can install `LightFM` through `pip`:

```
pip install lightfm
```

⚠ Some Ubuntu users may encounter problems during the installation process (*>> Failed building wheel for lightfm*, *>> Unable to execute 'gcc'*). In this case, you need first to install gcc compiler :

```
sudo apt-get update

sudo apt-get install gcc
```

2. Recommendation Data Preparation

Good luck in this new project! Open first notebook and follow instructions 🤓

3. Recommendation Model (OPTIONAL)

This is the second part of our recommendation challenge.

In this challenge you will **train the model** based on the data you prepared, and start **testing your recommendations**

Open the notebook and follow instructions.

4. Flask Application (OPTIONAL)

This is the third and last part of our recommendation challenge.

Today, we will deploy the movie recommendation engine we have been working on previously! To do so, we will use Flask framework (a Python library for web development), and "serve" our model to anyone!

The server we will use will be in local for now, but as soon as you are happy with your work, you can deploy it on the web and make it accessible to anyone (as a web-app for example).

First, start by installing Flask library:

```
pip install flask
```

And make sure you can launch the server on your machine. From `netflixApp` folder, run:

```
python main.py
```

And open `localhost:5000` in your favorite browser, you should successfully see the web app.

We built the skeleton of the app for you 😉 But you need to implement the recommendation part! This is the continuation of what you have been doing in the previous challenge.

1. Start by exploring the `netflixApp` folder. What directories and files does it contain. First, make sure to understand how the app is articulated and what are the different functions of the files.

2. In `utils/` folder, we wrote for you the functions used for loading the files in `load.py`. However, in `reco.py` you need to implement the functions `sims_to_recos`, `get_reco`. It should be quite similar to the ones you created yesterday.

   Read the comments associated to those functions and implement them. Test them (for example in previous notebooks, or in a new one, or in Flask by printing some logs).

3. In `server.py`, use the function created above for populating the variable `recos` containing the recommendations sent to the user.

4. In `templates/recommendations.html`, display the movies recommended to the user (recos computed in Flask backend) as `h4` text to the user.