# Project on Social Media Recommendation

## Background:

Social media recommendation plays a vital role in the era of information explosion. Recommender systems are crucial in fulfilling recommendation tasks and have significantly increased user engagement across various scenarios including e-commerce, micro-video recommendation, etc. They help users to filter out irrelevant information from a vast pool of web information and provide them with the content they are most likely to be interested in. In this project, we will explore recommendation algorithms for micro-video recommendation.

## Task Aim:

Students will be given an initial program to perform the basic recommendation on micro-videos. Specifically, given the historical user-item interactions and micro-video content features, the task is to extend the CF (Collaborative-Filtering) method to a content-based method to help improve the accuracy and support social discovery in micro-video recommendation.

## Initial Program and Dataset:

1. The implementation of a CF-based method based on matrix factorization (MF) [1] algorithm will be given.
2. A micro-video recommendation dataset will be given, including the training, validation, and testing sets.
    - The dataset includes user-item interactions, item visual features, and item category features.
    - An additional held-out testing set will be used during live evaluation.

## Enhancement to be Done by Students:

1. To extend the MF-based method to perform content-based micro-video recommendation.
2. To extend the MF-based method to support content discovery, e.g., encourage diversity in recommendation results.
3. To implement evaluation metrics for diversity evaluation.
4. To provide analysis of experimental results, e.g., improved accuracy, or the trade-off between recommendation accuracy and diversity.

## Evaluation:

We will assess the project from the following two aspects:

- **Algorithm exploration**: the implementation and presentation of the extended content-based algorithms.
- **Performance**: The accuracy & diversity of the results.

## Descriptions of Dataset:

**MicroLens Dataset**: this is a micro-video dataset collected from a micro-video sharing service in China. The data can be found in “data/”.

**Provided files:**

- **training_dict.npy**: the dictionary of the user-item interactions in the training set. For each key-value pair, the key is the user id, and the value is a list of interacted items of the user in chronological order.
  - For example: 0: [1,5,3,7]
- **validation_dict.npy**: the dictionary of the user-item interactions in the validation set. For each key-value pair, the key is the user id, and the value is the list of the ground truth item in validation set.
  - For example: 0: [6]
- **testing_dict.npy**: the dictionary of the user-item interactions in the validation set. For each key-value pair, the key is the user id, and the value is the list of the ground truth item in testing set.
  - For example: 0: [11]
- **visual_feature.npy**: the dictionary of the micro-video visual feature. For each key-value pair, the key is the item id, and the value is a 512-dimensional feature vector (numpy array) extracted by pre-trained deep learning models.
- **category_feature.npy**: the dictionary of the micro-video category feature. For each key-value par, the key is the item id, and the value is the anonymized category integer, e.g., 0: 5.

**Held-out testing set**:

We will randomly sample 100 users from the testing set for live evaluation. The held-out users will be removed from the validation set as well as the testing set but will be seen in the training set.

## Descriptions of MF Program:

MF program can be found in “code/”.

**Provided files**:

- **quickstart.txt**: document for a quick start of MF training and evaluation.
- **main.py**: training scripts, including codes for model creation, data preparation, training process, and evaluation.
- **test.py**: testing scripts, which load the saved model and evaluate the model on the testing set.
- **model.py**: core codes for the MF method, including initialization, and the forward process. The codes will be called from main.py or test.py.
- **data_utils.py**: some necessary codes for preparing the data for the PyTorch training framework. The codes will be called from main.py or test.py.
- **evaluate.py**: codes for evaluation and the evaluation metrics. The codes will be called from main.py or test.py.

[1] Steffen Rendle et al. 2009. BPR: Bayesian Personalized Ranking from Implicit Feedback. In UAI. AUAI Press, 452–461.