## Problem Statement

Most of the work we do at Bigtincan is around understanding user behaviour.

In a broad sense, Bigtincan Hub is a content management platform, where users are creating and interacting with their content.

One of the major problems we try to solve in our Data Operations team is suggesting right content to people based on their previous interaction.

## Sample Data

The file `data.csv` is a comma-separated file containing a random sample of user-item interactions. Both users and items are recorded with a unique id.

## The Application

Your coding challenge is to design and build a recommender system that will accept a specific user id and output top ten recommended items.

The application should also provide at least three expected performance metrics when run in production. You are free to come up with any performance measure, but provide a clear description of your selection.

## LIBRARIES and DEPENDENCIES

The following packages is used in this application and should be available on the runtime machine;

- numpy http://www.numpy.org/

- pandas https://pandas.pydata.org/

- scikit https://www.scipy.org/scikits.html

- scikit-surprise http://surpriselib.com/

## ASSUMPTIONS

In this implementation I assume that, the number of interactions between users and each item, shows the rate that user gives to that item. This rate is normalized to a limited range (1-5) to have a more realistic feature.

For each user-item couples, the timestamp of the most recent interaction is taken into account and the rest of them are ignored.

## DATA PREPARATION and CLEANING

For the reference model of this application, I have applied the following to the dataset:

- Set an initial rating feature (1) for each user-item interaction.

- Change the timestamp data format to unix\_timestamp format. (The file `Data\_rating\_timestamp.csv`)

- Ignoring time zone to simplify the data.

- Create pivot tables to aggregate data on rating column, compute the rate for each user-item couple and select the newest time for them.

- Normalize rating based on total interaction that the user had.

- Scale rating to 1-5.

- Generation of final csv file 'data\_rating\_timestamp\_normalized.csv'

## DATA LOADING

Just before the learning processes, data is loaded from a file called 'data\_rating\_timestamp\_normalized.csv'. The file is generated during runtime after pre-processing the original data file.

## PROCESS (LEARNING and PREDICTING)

- This code offers 4 recommendation algorithms based on Collaborative Filtering. User will be asked to choose which algorithm to proceed with during runtime.

- Also offers 3 splitting data methods for train and test, which is again selectable by user.

- Based on options selected by user (operator), the model is built, the rates are predicted, the top n recommender items for each user is founded and the performance metrics are computed.

- Then the application will wait for the operator input to enter user-id. This user-id will be sent to the prediction engine and the result will be shown.

- Finally, the operator is provided with an option to run 5-fold cross-validation based on the selected model and print results.

## Performance Metrics

I used MAE and RMSE values that actually determine how good the system performed at predicting the rating value for unknown items.

- MAE (Mean Absolute Error) computes the average error. We use the absolute value for each error.

- RMSE (Root Mean Square Error): computes the average error too, except that instead of taking the absolute value we square the errors.

This has the effect of having only positive errors

Precision and Recall measure that determine how good the model were at the predicting the top n recommendation for users.

- Recall: What ratio of items that a user likes were actually recommended?

- Precision: Out of all the recommended items, how many the user actually liked?

## HOW TO USE THE PACKAGE

Note: In order to run the package in python you need "scikit-surprise" package, which is mentioned in "LIBRARIES and DEPENDENCIES" section

n. (http://surpriselib.com/)

-Install Surprise package by one of these ways:

- With pip:

pip install numpy

pip install scikit-surprise

- With conda:

pip install scikit-surprise” in Anaconda Prompt

- Install package in Anaconda Navigator Environments in channel conda-forge

-Extract content from RS-Project zip file.

-Open RS\_main python file and execute it. (I used Spyder IDE)

-After the executing, multiple options are shown to the user (about to choose which algorithm to proceed) and the results will be printed.