Recommender System Project

Tatiana Shingel Moody

March 23, 2014

1 Introduction

This is a summary of the project and some details of the code. The code is written in Python.

The following files are provided:

- main_project.py is the Python module containing all functions necessary to implement histogram plots and recommendations. The main_project.py is built in the form of a simple API. This is a summary of the commands it accepts through standard input:
  - Histogram
  - Pairs Histogram
  - recommend-likes likes="LikeString1,LikeString2,etc..."
  - recommend-users likes="LikeString1,LikeString2,etc..."
- A csv file of "anonymized" Facebook **Like** and **Interest** data. Each row in the data file is of the format: **UID**, **Like1**, **Like2**, etc. The size of the file is approximately 1GB. The data is noisy with lots of likes in foreign languages.
- UserLikesPrint.py is the Python module which outputs User-Likes data to the console row-by-row after reading and cleaning the data from the above csv file. It is included for convenience and can be used to assemble lists of Likes data to test the code.
- A txt file of most common English words.

2 Data Analysis

2.1 Data Cleaning

The data is cleaned with the help of the provided file of English words. Several data structures are used in processing the data. All UIDs are stored as strings in a list. The position of each string UID in the list is an integer UID which is a unique identifier for this string. All Likes are also stored in another list and numbered by their positions. A hash table is used to store lists of Likes unique identifiers with keys being integer UIDs. Another hash table has Like strings as keys and Like integer identifiers as values.

2.2 Histogram

Command line input “Histogram” generates a histogram of Likes data. That is, a graph is produced that displays Likes on the x-axis and frequency count on the y-axis. At most ten most frequent Likes are displayed for clarity.

2.3 High-Frequency Pairs

Command line input “Pairs Histogram” generates histogram of high-frequency Like pairs. The produced graph has Like pairs on the x-axis and frequency count on the y-axis.

3 Recommender Systems

3.1 Recommending Likes Based on the Given Set of Likes

Command line input:

```
>recommend-likes likes=“LikeString1,LikeString2,LikeString3”
```

The recommendation is implemented according to the following steps:

- **Computing similar list of items for a given item** I_0: collect all Users who like I_0 and take the union of each User’s item list. Let’s call this list A. Next, discard I_0 from A. Return a hash table with keys as items from A and values as similarity scores between I_0 and every item in A.
- **Similarity score between two items**: look at two sets of all users who like each of the items. Compute Jaccard index between these two sets.
- **Computing similar list of items for a given list of items**: given the list of items A, for every element in A apply function **similarity_for** which returns a hash table of similar items together with their similarity scores. Generate a python dictionary (hash table) having similar items as keys and lists of similarity scores as values. Then aggregate each list of similarity scores by taking a mean.

3.2 Recommending UIDs based on the given set of Likes

Command line input:

```
>recommend-users likes=“LikeString1,LikeString2,LikeString3”
```

The recommendation is implemented according to the following steps:

- Collect all Users who like items from the given set of Likes and save them to a list of Users, say U_0. Take all those User’s Likes and save them to a list of Like data, say B. For B, compute a similar list of Likes using the recommendation process 3.1. Assemble a new list of Likes containing B and similar Likes with high enough similarity scores. For this new list of Likes, collect all Users who like the items from the list and save them to a list, say U_1.
- For every User in U_1, assemble a list of Users who like items from this User’s item list. Compare this list of Users to U_0 using Jaccard index. Return a hash table with Users from U_1 as keys and similarity scores as values.