COMP3208 - Social Computing Techniques
Module Assignment

Azzino Tommy & Drago Matteo

Theoretical Introduction

A recommender system is a technology that involves two side of the same coin: *users* and *items*. Items could be products, movies, events, articles that are going to be recommended to users such as: customers, app users, travellers. We can relate the concept of recommendation to a real life situation: if you think to a small shop or boutique, then it's usual that a merchant knew personal preferences of everyday customers; his high quality advices satisfy clients increase profits and visibility of the shop (thank also to the usual word-of-mouth between people).
When dealing with online market places (Netflix, Amazon, Ebay or Asos), personal recommendations, suited up for each particular and different user, can be generated by something that can be seen as an "artificial merchant": the **recommender system**.
There are several definitions that we can find in the literature and in the *Internet*. Quoting Wikipedia:
"A recommender or recommendation system (sometimes replacing "system" with a synonym such as platform or engine) is a subclass of information filtering system that seeks to predict the "rating" or "preference" that a user would give to an item".
In our opinion such definition is not completely correct; recommender systems are more than what stated. We can think about it as an *personal assistant*, because it can help you in order to make the right decision when you're going to buy something. Moreover, it knows your preferences and tastes; therefore it's not only a machine or an IT system: it perfectly suits you. A more proper definition could be:
"A recommender systems is a system that help users discover items they may like, based on a sort of personal knowledge of each user".
Formally, such systems filter different items after studying a personal profile. They differ from many other information filtering systems, such as standard search engines and *reputation systems*, because recommendations are made on the specific user or group of users. We want to stress that the main goal for a recommender system is to show the user what he may like, guiding him on a platform that potentially offers a wide and heterogeneous set of choices. To achieve this purpose, the system can adopt different *user* or *item-based* models such as: the given rates to different items, preferences on distinct items typologies, either *demographic* or *context* information.
After the design of a good model, we need to find a way to elaborate this huge amount of information in order to extract the peculiarities between users and items that allows to achieve good recommendations.
Depending on the objective you want to fulfill, there are many different recommendation paradigms. The choice of the best paradigms is to be found according the type of data that is available and the computational efforts that one would like to employee when building the recommender system. For the latter point, computation time and power of available resources are two to important factors to be considered when developing such systems. In addition, when talking about the tyoes of data available, a good idea to use hybrids systems that combines multiple different approaches to address each method's limitations. Nevertheless, it is worthwhile discussing the different paradigms in order to understand how to efficiently combine them.
There are four main paradigms in recommendation systems, which are described below:
content-based: content-based recommendation builds a user's profile that is automatically generated and updated using, for example, items already bought by the user. Such algorithms recommend similar items based on item description, looking at items that are similar to each others according to certain particular features. The main advantage of content-based recommendation is that it does not require too much past history regarding one user, and among users and items. In many everyday scenarios, content-based recommendation is the most common approach. For example, personalized search is achieved by recommending web pages which are similar in content to those previously viewed by an user.
knowledge-based: knowledge-based recommender systems use manually generated information about both users and items, that are built asking explicitly to a user questions about its preferences in order to construct a personal profile. Then, item similarity is done by experts. This is one limitations of this recommendation paradigms: we need the presence of experts in the sector capable of determine the best recommendation for

a user. Nonetheless, when recommending to first-time users, knowledge based becomes one of the strategy that gives best performances.
social and demographic: these recommender systems suggest items that are liked by friends or friends of friends, or items liked by demographically-similar people. Furthermore, they don't need any preferences by the user, making them very powerful and easy to apply.
collaborative filtering: collaborative filtering is the most popular technique involved in the construction of recommender systems. This approach is used by many companies and e-commerce sites. The idea behind is to build a matrix that represents similarity between users or items. This is then use to predict missing preferences for a certain item for example, and recommend items with the highest predicted rate to an user. The straightforward advantage of this approach is that there's no need of content information for an item or the presence of experts that suggest a particular item. Only a set of users and items together with a notion of "preference" are needed to implement collaborative filtering. Moreover, this technique has been studied and researched by an huge amount of experts, making it pretty well-understood.
In our work, we concentrated on the development of a recommender system based on collaborative filtering approach, with details that are given in the following sections.

MATLAB Approach

Our first idea was to evaluate the similarity matrix with MATLAB, given that it is naturally optimized for operations with matrices and vectors.

C++ approach

Considering the problems encountered with MATLAB, we moved the development of the recommender system in C++ environment. The steps, involved in the construction, are reported in details in the remaining of this section.

Database management

Considering the huge amount of users' ratings contained in the *csv* file, we decided to utilize *sqlite* as a database where to store all information needed and easily retrieve them. Each row of the database's table consists of three entries: the first contains user's id, the second item's identification while the third represents the rate given to that particular item by the user. In order to speed up database access several indexes were created on this database. An index called *users_idx* was set up for fast retrieving of data by user id, while an index called *items_idx* has the same function, but it's used with items search based. Another index (*itemsusers_idx*), it's used for researches based on a double key in the form of user and item ids. When dealing with big databases like that, the creation of an index is really important and can speed up database access of almost 100 times, comparing with no indexes case.
Moreover, as in **??** need the average of the rates given by a user to some item, we constructed a database that contains this values. Thanks to that, we don't need to recompute thousands of time the same value. Obviously, also this database contains an user index in order to speed up access.
The similarity matrix will be stored in database formatted in this way: each row contains within the first two columns a couple of ids that identify two users while the third columns contains the amount of similarity for the pair. An index between column 1 and 2 will be created. This index is very important since the database storing similarity values has a disk size of ≈ 60 GByte.
The last but not least database to be used is the one that will contain the predicted rates given for an item by a user. The first two columns of this db are formatted according to *csv* file *"comp3208-2017-test"* and the third will be the predicted rating.

Building similarity matrix

As explained above we started by considering user-based collaborative filtering. In order to apply this paradigm, we need to build a similarity matrix between users. The equation that gives *Paerson coefficient* for similarity is given in **??**.
The procedures that involve the construction of the similarity matrix, exploiting *sqlite* in C++ environment, are the following:

- primo elemento
- secondo elemento