Query recommendation system

Eros Ribaga
University of Trento
eros.ribaga@studenti.unitn.it

Simone Compri
University of Trento
simone.compri@studenti.unitn.it

1 INTRODUCTION

In today's digital environment, it is practically impossible for individual users to properly search and find any sort of material, which can range from videos on Youtube to papers about some specific subject, without the aid of recommender systems due to the enormous amount and diversity of data.
As a consequence of the increasing amount of available data online, recommendation systems have become increasingly significant as they assist companies and organizations in personalizing user experiences and enhancing consumer engagement.
Without recommendation systems, user experiences would feel stressful and disorienting, as it could take users hours and hours to find something that can presumably fit their interests, maybe to find out that their search has come to an unsatisfactory result.
Now, companies rely on recommendation systems to make their services satisfying for the ones utilizing them, by granting a user experience that can make users attached, or loyal, to the company.

Recommendation systems depend on algorithms that suggest items or actions to users based on their interests and past behaviours.
They typically operate by collecting data on users and the items they interact with and then using this data to identify patterns and make personalized recommendations.
There are several different types of recommendation systems, including:

- collaborative filtering
- content-based filtering
- hybrid systems that combine both approaches
- clustering
- and many more

Disclaimer: as our solution and experiments are still to be well defined, the next two paragraphs are a work in progress.

In this report, we will provide an overview of a problem that uses this system and its solution.
More precisely, the problem just announced consists of a recommendation system that will work in a situation where there are users who can ask questions to retrieve information about people and then give explicit feedback by rating how satisfied they were with the result.
The solution that we found consists of using an item-item collaborative filtering approach of recommendation systems and a preprocessing of the queries, using an algorithm based on Frequent Itemset, to keep track of the similarity between them.

Afterwards, we will perform experiments to prove that our solution is valid and competitive with the respect to the other proposed online or by other groups of the data mining course. Some examples consist in executing our solution and

- comparing its result to the expected one, contained in a filled utility matrix generated alongside the dataset
- comparing, with respect to our and others' generated datasets, our solution's result and execution time on with the ones of:
  - other group solutions
  - online solutions

Unlike most situations, the problem we are trying to solve expects users to give explicit feedback on the result of queries, instead of basic items.
A **query** is defined as the way in which users are able to interact with the system. In fact, once a user poses a query to the system, the latter will return a set of objects that should correspond to what the user is presumably looking for.
This adds an intermediary step to the system, in fact, the utility matrix only represents the appreciation of users toward a subset of objects, instead of the object itself.
Implementing a recommendation system for this problem leads to several challenges, like managing a big amount of data or the scalability of the solution.
But, the main challenges this project encounters are in particular the ones that involve queries. In fact, they add a layer of complexity to the problem by adding some complications:

- it's harder to tell whether a user is interested or not in a singular object
- the same subset of objects could be voted more times by the same user, as two or more queries, that could have either an identical definition (but different ids) or even a different one, could generate the same result set

2 PROBLEM STATEMENT

Before exploring in detail how this problem was solved, a formal definition of the latter is given in this section.

The issue that we are attempting to resolve is trying to fill a sparse user-query matrix, given in input four sets of objects:

- **Users:** a set of unique IDs $U = \{user_1, user_2, \ldots, user_n\}$, each one representing a different user
- **Objects:** a set of objects $O = \{o_1, o_2, \ldots, o_k\}$, where each object is characterized by some defined features; this means that every feature is described by a meaningful value
- **Queries:** a set of queries $Q = \{q_1, q_2, \ldots, q_m\}$ that have been posed in the past, each of them characterized by a unique query id and its definition, i.e., a conjunction of a set of conditions, each one referred to a feature
- **Utility matrix:** a sparse user-query matrix $UM \in R^{n \times m}$ that links every user-query tuple to a value

from 1 to 100 (**score**), which indicates how satisfied that user is with the answer of that certain query.

To better understand how the relationship between users and queries works, it is appropriate to introduce the key elements that are going to be treated.

A **query** is defined as the way with which the users are able to interact with the system, once a user poses a query to the system, the latter will return a set of objects that should correspond to what the user is presumably looking for.
The result of every query is then evaluated by the user with a **score**, which indicates how happy he was with the answer to that certain query.

The **Utility matrix**'s purpose is to collect all the scores given by users to the queries.
To put it another way, we will have a rating matrix, where each row represents a user's rating for each object and each column represents all user ratings for a single object. Our task is to fill in all the blank cells in this matrix using the data from the initial matrix.

To sum up, our problem can be described as a **function** that, given as input a list of objects O, a set of users U, a set of queries Q, and a sparse utility matrix UM returns the latter filled up by all scores.

All of this is done to develop a recommendation system that suggests a set of queries to the user based on the expected scores; answer sets should include information the user is interested in. In more detail, this means that the algorithm must fill UM in a way that the predicted scores are as close as possible to the scores that the users would provide to the different queries.

The **Second part** of the problem asks, given as input a query Q and the filled utility matrix UM, a way to compute the importance of the input query.
The **importance** of a query is defined as a metric that could be computed with respect to the whole dataset, like the number of times a query very similar to Q has been posed, or regarding also the utility matrix, like the average scores of all the queries that are related in some way to Q. However, this concept will be developed more in the next sections of the report.

So, defining it formally, this part of the problem can be described as a function, that takes the filled UM and a query as input, and returns a value (that can be expressed as a percentage, a qualitative evaluation, etc.) that describes how important the query with respect to the utility matrix and the whole dataset.