Scaling a recommender system across large data volumes

# Building a recommendation engine in presence of large data volume

E-commerce businesses can suggest new products to their customers. How to they chose the products to recommend? The companies collect data about past purchases of their customers. Starting from that, they can identify items that have been purchased by the same customers and customers that purchased the same items. Given a new customer, we can recommend him/her items similar to the ones that he/she purchased already, or items that have been purchased by similar customers. There are also other data sources like personal information, descriptions of the items, reviews. For the sake of simplicity, in this article we deal with information about purchases only.

Large companies like Amazon and eBay already built sophisticated recommendation engines tailored to their environment. What if a smaller company wants to build a smaller recommendation engine from scratch? The context is much more simple although the volume of data to process may still be large.

In this article, we show how to build simple recommender systems that can be scaled across large data volumes. Due to the complexity of the problem and of the variety of contexts, the article describes it from a high-level perspective.

# Choosing the environment

In this article, the technology that we use is SQL Server 2016 with R Services. The reason is that we can store large SQL tables and process them using advanced analytics techniques provided by R.

The dataset is about movies. Given a set of movies and users, the table describes what movies each user has watched. To avoid further complication, the table doesn’t include user ratings. Its structure is:

* A row for each movie
* A column for each user
* Cells equal to 1 if the user watched the movie and to 0 otherwise

The data has already been imported into SQL 2016 into a table called *movies*. To analyse the data, we use the R environment and we define an object called *sql\_movies* that allows us to access the SQL table.

CODE

Starting from *sql\_movies*, we can perform out analyses. The R package *recommenderlab* contains pre-built tools to build a recommender system. However, the package need all the data to be on the RAM, so it can deal with relatively small data volumes only. Therefore, dealing with a larger dataset, we need to reduce its volume first.

# Reducing the number of users

A common challenge is that there are many users that in our dataset correspond to columns.

A solution consists is to identify small groups of similar users and to define a new table having a column for each group of users. The steps are

1. Measure the similarity between users using the function rxCor
2. Starting from the similarity, define a matrix containing a distance between the users
3. Starting from the distance matrix, identify groups of users applying hierarchical clustering
4. For each group of users, compute a column containing the number of times they watched each movie

The new table contains a row for each movie and a column for each group of users.

# Reducing the number of items

If there is a large number of items, the table will contain a lot of rows. Using an approach similar to the users, we can identify groups of similar items and define a table with a row for each group of items. For that purpose, we can just use the function rxKmeans.

# Building the engine

After having reduced the size of the data, we can pull in-memory the small-sized table. Then, we can use the package *recommenderlab* to build a recommendation engine.

Build on small matrix

Apply to clusters of movies

Apply to clusters of users

Measure performance

# Conclusions