Rating Prediction for Yelp Recommendation System

Final Project Report

Arthur CLAUDE
CentraleSupélec
Gif-sur-Yvette, France
arthur.claude@supelec.fr

Armand MARGERIN
CentraleSupélec
Gif-sur-Yvette, France
armand.margerin@supelec.fr

Louis TERNON
CentraleSupélec
Gif-sur-Yvette, France
louis.ternon@student.ecp.fr

Thomas DELRUE
CentraleSupélec
Gif-sur-Yvette, France
thomas.delrue@student.ecp.fr

ABSTRACT

This document presents the outline of the project we conducted in the frame of the Network and Graph-based Science Analytics course at CentraleSupélec. This project is focused on the implementation of a restaurant recommendation system. The main step in such a system is the prediction of the rating value that a user will give to a product. For that purpose, many methods have been proposed. We implemented a number of models that we found interesting and compared their performance on a dataset built from the open Yelp Dataset.

CCS CONCEPTS

• **Mathematics of computing → Graph theory**.

KEYWORDS

Network and Graph, Recommendation System, Rating Prediction

ACM Reference Format:
Arthur CLAUDE, Armand MARGERIN, Louis TERNON, and Thomas DELRUE. 2020. Rating Prediction for Yelp Recommendation System: Final Project Report. In *CentraleSupélec '20: Network and Graph-based Science Analytics, March 2020, Gif-sur-Yvette, FR.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION - MOTIVATION

With the continuous growth of the web, the size and complexity of many websites grew up. It is sometimes difficult and time-consuming for users to find the information they are looking for. Moreover, users do not always know what they want beforehand and their interests may change over time.
A Recommender System is a process that seeks to predict user preferences in order to suggest relevant items. They have been widely applied to eCommerce and personalized recommending services, and are now present everywhere on the web. For example, Amazon uses it to suggest products to customers and Netflix uses it to provide personalized movies recommendation to each user.

Recommendation systems can be really critical in some industries as they can improve profits when they are efficient. For example, personalized recommendations can help increase sales in eCommerce and improve the users' experience with services (key to enhancing user satisfaction and loyalty).

A large number of methods have been developed in order to implement recommendation systems. In most cases, the algorithm predicts the numerical ratings that users give to products and uses this information to make recommendations. In our project we are interested in the rating prediction step of recommendation systems.

The different methods proposed over time each have strengths and weaknesses. In this project, we compare some of them on a particular dataset.

2 PROBLEM DEFINITION

In order to evaluate the selected methods, we will use a dataset [1] provided by the website Yelp. This dataset is a subset of their businesses, reviews, and user data. The file contains six data files described below:

- business.json: contains business data including location data, attributes, and categories.
- review.json: contains full review text data including the user_id that wrote the review and the business_id the review is written for. This data file contains 7 million reviews from 1.6 million users across 190 thousand business in 10 metropolitan areas.
- user.json: contains user data including the user's friend mapping and all the metadata associated with the user.
- checkin.json: contains the checkins on a business.
- tip.json: contains tips written by a user on a business. Tips are shorter than reviews and tend to convey quick suggestions.
- photo.json: contains photo data including the caption and classification (one of "food", "drink", "menu", "inside" or "outside").

CentraleSupélec '20, March 2020, Gif-sur-Yvette, FR

ACM ISBN 978-1-4503-XXXX-X/18/06…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

For our project, we focus on the first three data files presented above.

The entire dataset is really large. In order to make the implementation of the different methods lighter and faster, we built a smaller dataset.

In this way, we firstly remarked that users can rate a business more than one time, so we kept only the last one. Then, we decided to work only with restaurants of the Illinois State. For that, we selected businesses that have «restaurant» or «food» in their descriptions, and are located in the Illinois State. We ended up with 840 restaurants, and finally, we decided to keep a sample of 3000 users.

Our final dataset is composed of three files:

- businesses_restaurants_illinois_reduced.csv: contains business data including location data, attributes, and categories for the selected 840 businesses.
- ratings_restaurants_illinois_reduced.csv: contains ratings data including the user_id that wrote the review and the business_id the review is written for. It contains 22520 ratings.
- users_restaurants_illinois_reduced.csv: contains user data including the user's friend mapping and all the metadata associated with the user, for the 3000 selected users.

We finally split the ratings data into a train set and a test set thanks to the dates. We keep for testing the last rating of each user. It is indeed this separation which seemed closest to the real functioning of a recommendation system. Moreover, some users of the dataset have only one rating, they thus only appear in the test set, and will allow us to evaluate our system's ability to handle new users.

We implemented our rating prediction systems and compared their performance on this particular dataset. This comparison is presented in the remainder of this report.

3 RELATED WORK

Over the last two decades, researchers have worked extensively to find the best way to recommend products to users.

It is possible to build a weighted bipartite graph containing on one side all the users and on the other side all the restaurants. The edges of this graph are thus the ratings. Several methods exploiting the information contained in this graph allow to predict the new ratings.

The first methods we are going to work on are the well-known methods of Collaborative Filtering [4]. Collaborative Filtering is one of the most popular recommendation systems strategies. The main purpose of these algorithms is to analyze relationships between users and inter-dependencies among products in order to predict the rating a user will give to an item [7].
There are two main areas of Collaborative Filtering: Neighborhood Methods and Latent Methods.

Neighborhood methods are centered on computing the relationships between users, or between items [5]. These methods are based on the idea that people who agreed in their evaluation of certain items in the past are likely to agree again in the future. User-based methods measure the similarity between target users and other users while item-based methods measure the similarity between the items that target users rate and other items.
On the other hand, Latent Methods create a latent feature to compare one user to another one. Some of the successful realizations of Latent Methods are based on matrix factorization [5]. For example, the SVD algorithm is well known as it has been popularized by Simon Funk during the Netflix Prize [3]. Finally, factorization methods based on Deep Learning have been proposed and lead to interesting results [2].

As a next step, we will study a bipartite network projection method [9] that use one-mode projecting and a particular weighting method to extract the hidden information of networks. According to the publication, this method can significantly outperform Collaborative Filtering in performance.

Then, we will use a particular feature of the dataset used in order to set up a particular model. In fact, on the Yelp network, users can built their explicit social network by adding each other as friends. It is possible to build a unipartite graph containing all the users, and to calculate new similarities between users in order to implement a new collaborative filtering algorithm.

Then, it is possible to implement a recommendation methods that takes into account the two graphs introduced above. This kind of method has been presented [8] and yields interesting results.

Finally, as the rating is a quantitative variable we want to predict for a couple (user,business), it is possible to see the problem as a regression problem. Different regression models in Machine Learning have already been developed [6].

4 METHODOLOGY

As explained earlier, from our dataset, it is possible to build a weighted bipartite graph containing on one side all the users and on the other side all the restaurants. The edges of this graph are the ratings of the training part of the dataset. Our goal is then to predict the weights of the edges provided in the testing set from the information contained in the graph. We can also build a user unipartite graph based on the explicit social network of the Yelp website. We used these two graphs to build rating prediction methods.

The different existing methods have each strengths and weaknesses. In our project we focused on the following methods:

- Rating Prediction Based on User-Item Bipartite Graph:
  - User-based Collaborative Filtering
  - Item-based Collaborative Filtering
  - Latent Collaborative Filtering by SVD and SVD++
  - Deep Matrix Factorization Collaborative Filtering
  - Bipartite Graph Projection Method

- Rating Prediction Based on User-User Unipartite Graph:
  - User-based Collaborative Filtering
- Rating Prediction Based on Both Graphs
  - User-based Collaborative Filtering
- Rating Prediction by Regression

In the following, we note the users ensemble U and the restaurants ensemble I. The rating given by the user u to the restaurant i is noted $r_{u,i}$.

4.1 Rating Prediction Based on User-Item Bipartite Graph: User-based Collaborative Filtering

In order to set up a user-based Collaborative Filtering method, it is necessary to compute similarities between users of our bipartite graph. In our case, we used the cosine similarity. For two users u and v of U, this similarity is:

$$sim(u,v) = \frac{\sum_{\forall i \in I}(r_{u,i} r_{v,i})}{\sqrt{\sum_{\forall i \in I}(r_{u,i})^2}\sqrt{\sum_{\forall i \in I}(r_{v,i})^2}}$$

By using this formula, we can build a similarity matrix between users, and compute ratings predictions thanks to a weighted average. The next formula is the one used for a given user u and a restaurant i.

$$\hat{r}_{u,i} = \frac{\sum_{\forall v \in U}(sim(u,v) r_{v,i})}{\sum_{\forall v \in V} sim(u,v)}$$

For « unknown » users, for which the similarities with other users are all zero, the calculation leads to a division by zero. In this case, the predicted rating is set to the average rating of the train set.

4.2 Rating Prediction Based on User-Item Bipartite Graph: Item-based Collaborative Filtering

This second method is similar to the first one, except that similarities are calculated between restaurants. The used formulas are the following.

For two restaurants i and j of I, the similarity is:

$$sim(i,j) = \frac{\sum_{\forall u \in U}(r_{u,i} r_{u,j})}{\sqrt{\sum_{\forall u \in U}(r_{u,i})^2}\sqrt{\sum_{\forall u \in U}(r_{u,j})^2}}$$

By using this formula, we can build a similarity matrix between restaurants, and compute ratings predictions thanks to a weighted average. The next formula is the one used for a given user u and a restaurant i.

$$\hat{r}_{u,i} = \frac{\sum_{\forall j \in I}(sim(i,j) r_{u,j})}{\sum_{\forall j \in I} sim(i,j)}$$

As in the previous part, for « unknown » restaurants, for which the similarities with other restaurants are all zero, the calculation leads to a division by zero. In this case, the predicted rating is set to the average rating of the train set.

4.3 Rating Prediction Based on User-Item Bipartite Graph: Latent Collaborative Filtering by SVD and SVD++

Matrix factorization models map both users and items to a joint latent factor space of dimensionality f, such that user-item interactions are modeled as inner products in that space. Accordingly, each item i is associated with a vector $q_i \in \mathbb{R}^f$ and each user u is associated with a vector $p_u \in \mathbb{R}^f$. The dot product, $q_i^T p_u$ captures the interaction between user u and item i. This approximates user u's rating of item i:

$$\hat{r}_{u,i} = q_i^T p_u$$

Additionally, much of the observed variation in rating values is due to effects associated with either users or items, known as biases, independent of any interactions. Thus, it would be unwise to explain the full rating value by an interaction of the form $q_i^T p_u$. A first order approximation of the bias involved in rating $r_{u,i}$ is:

$$b_{u,i} = \mu + b_i + b_u$$

Where μ is the overall average rating, and b_i and b_u indicate the observed deviations of user u and item i from the average.

The new rating prediction formula is:

$$\hat{r}_{u,i} = \mu + b_i + b_u + q_i^T p_u$$

If user u is unknown, then the bias b_u and the factor p_u are assumed to be zero. The same applies for item i with b_i and q_i.

The major challenge is to compute all the vectors $q_i, p_u \in \mathbb{R}^f$. Such a model is related to Singular Value Decomposition (SVD), a well-established technique for identifying latent semantic factors in information retrieval. In the collaborative filtering domain, applying SVD requires factoring the user-item rating matrix. This can raise difficulties because of the high portion of missing values in the user-item ratings matrix. Recent works suggest modeling directly the observed ratings only, while avoiding overfitting through a regularized model.

In this model the system minimizes the following regularized squared error on the set of known ratings in order to estimate the unknown:

$$min \sum_{(u,i) \in K} (r_{u,i} - \hat{r}_{u,i})^2 + \lambda(b_i^2 + b_u^2 + ||q_i||^2 + ||p_u||^2)$$

where K is the set of the (u,i) pairs for which $r_{u,i}$ is known, and λ is determined by cross-validation.

The minimization is performed by a very straightforward stochastic gradient descent:

$$\begin{aligned} b_u &\leftarrow b_u + \gamma(e_{u,i} - \lambda b_u) \\ b_i &\leftarrow b_i + \gamma(e_{u,i} - \lambda b_i) \\ p_u &\leftarrow p_u + \gamma(e_{u,i} \cdot q_i - \lambda p_u) \\ q_i &\leftarrow q_i + \gamma(e_{u,i} \cdot p_u - \lambda q_i) \end{aligned} \tag{1}$$

Where $e_{u,i} = r_{u,i} - \hat{r}_{u,i}$. These steps are performed over all the ratings of the train set and repeated a given number of epochs. b_u and b_i are initialized to 0, and q_i and p_u are randomly initialized according to a normal distribution.

The *surprise* library in Python allows to use this method.

Finally, the SVD++ algorithm is an extension of SVD taking into account implicit ratings. The new rating prediction formula is:

$$\hat{r}_{u,i} = \mu + b_u + b_i + q_i^T \left(p_u + |I_u|^{-\frac{1}{2}} \sum_{j \in I_u} y_j \right)$$

Where the y_j terms are new set of item factors that capture implicit ratings. Just as for SVD, the parameters are learned using a stochastic gradient descent on the regularized squared error objective.

4.4 Rating Prediction Based on User-Item Bipartite Graph: Deep Matrix Factorization Collaborative Filtering

This method is based on the work of [2]. It uses a sparse user-item matrix interaction, and neural networks to estimate the cosine similarity between a user and a restaurant. From this estimation results the final mark.

Suppose there are M users U = $\{u_1, u_2, ..., u_M\}$, and N items I = $\{i_1, i_2, ..., i_N\}$.
Let $R \in \mathbf{R}^{M \times N}$ be the rating matrix, where $R_{m,n}$ denotes the rating given by individual u_m to restaurant i_n. We denote the user-item matrix Y as follows :

$$Y_{m,n} = \begin{cases} 0 & \text{if } R_{m,n} \text{ is unknown} \\ R_{m,n} \in [\![1\ ,\ 5]\!] & \text{otherwise.} \end{cases}$$

$Y_{*}n$ denotes the columns n of Y, and $Y_{m}*$ the row m of Y.

We end up with a sparse matrix where all non-zero cells are the ratings the users have already given. The matrix R will act as our training set for the construction of our model.

Recommender systems are very often formulated as the problem of estimating the rating of each unobserved entry in Y. Model-based approaches assume all ratings can be generated as follows :

$$\hat{Y}_{m,n} = F(u_m, i_n | \Theta)$$

where $\hat{Y}_{m,n}$ denotes the predicted rating of interaction $Y_{m,n}$ between user u_m and restaurant i_n. Θ denotes the model's parameters and F the function that maps parameters to the estimated rating. The main difficulty is to define the function F or, which is tantamount, come up with a model. Latent Factor Model (LFM) [5] applied the dot product of p_{u_m} and q_{i_n} to predict $\hat{Y}_{m,n}$, where p_{u_m} and q_{i_n} are latent representations of u_m and i_n.
The Deep Matrix Factorization Model [2] aims at building new latent representations p_{u_m} and q_{i_n} with neural networks. For an estimation $\hat{Y}_{m,n}$, the model uses two separates inputs Y_{*n} and Y_{m*} that are fed separately to two fully connected neural networks. In our experiments, both neural networks had two layers, with ReLU activation. User's intermediate layer was in dimension 1024 and restaurant's intermediate layer was in dimension 512. Both neural networks output a 64-dimensional tensor. Cosine similarity is then computed to give the output $\in [0, 1]$. For validation, the output was then multiplied by 5 and rounded to be consistent with other restaurants' ratings.

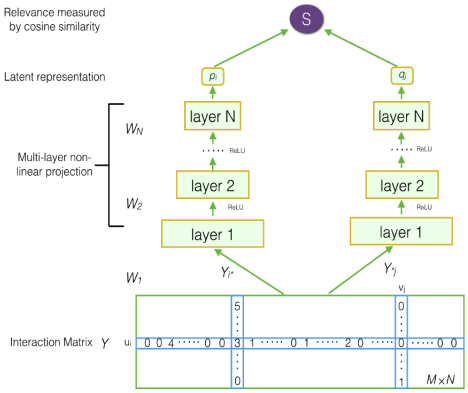


Figure 1: Model's Latent Factorisation with neural networks

To be trained, neural networks try to optimise a loss function. In this case, we can find inspiration from binary cross entropy.

$$L := -\sum_{m,n} (Y_{m,n} \log(\hat{Y}_{m,n}) + (1 - Y_{m,n}) \log(1 - \hat{Y}_{m,n}))$$

It is usually used for binary classification problems, where $Y \in \{0, 1\}$. In our problem however, $\hat{Y} \in [\![1\ ,\ 5]\!]$. As a consequence, we tweaked the loss function as :

$$L := -\sum_{m,n} (\frac{Y_{m,n}}{5} \log(\hat{Y}_{m,n}) + (1 - \frac{Y_{m,n}}{5}) \log(1 - \hat{Y}_{m,n}))$$

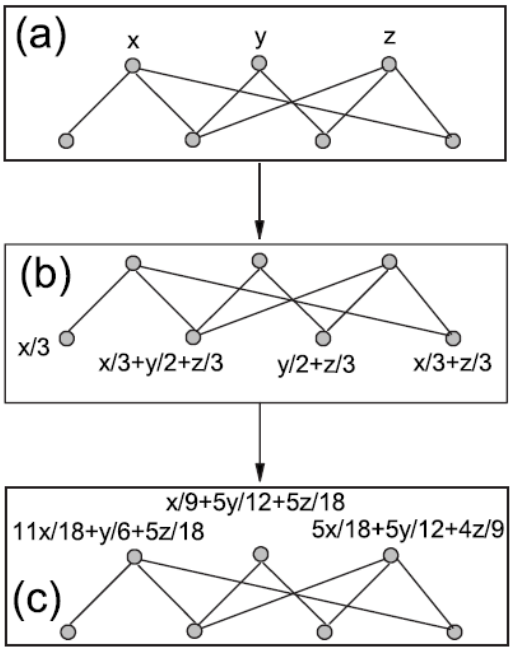
5 is the max rating a restaurant can get. It acts a a normalisation factor.

4.5 Rating Prediction Based on User-Item Bipartite Graph: the Network Based Inference

4.5.1 Motivation. As explained earlier, the system can be represented by a bipartite user-restaurant graph. Such graphs are usually dealt with one-mode projections. Traditional weighting methods cannot compensate the information loss incured when projecting on a single node set. Indeed, the weights are always symmetric and self-connection is not allowed.

To tackle this problem, Zhou et al. proposed an innovative weighting method for unimodal projections. In [9], they explain why their "Network Based Inference" should be applied to personal recommendation. We implemented the article approach and tested it on the Yelp data.

4.5.2 Weighting principle. For the sake of clarity, we will take a toy graph first and extrapolate to the Yelp dataset later.



Network Based Inference (NBI) is a resource allocation algorithm:

(a) We first provide the upper nodes with resources x, y and z.
(b) We propagate the resources to the lower nodes. Each upper node sends the same amount of resource its adjacent lower nodes. For instance, the 2^{nd} lower node is connected to all the upper nodes. Upper degrees are respectively 3, 2 and 3, therefore the 2^{nd} lower node receives a total resource $\frac{x}{3} + \frac{y}{2} + \frac{z}{3}$.
(c) We send the new resources back to the upper nodes.

In our toy example we have:

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} 11/18 & 1/6 & 5/18 \\ 1/9 & 5/12 & 5/18 \\ 5/18 & 5/12 & 4/9 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

We can see that this matrix is not symmetric and that its diagonal coefficients are all positive.

Back to the general case, let us consider $G = (R, U, E)$ a bipartite graph with node sets $R = (r_1, ..., r_n)$ and $U = (u_1, ..., u_m)$. We want to project G on the node set R and to weight the resulting projected graph.

Let $f_0 = (f_0(r_1), ..., f_0(r_n))$ the initial resources array on R, $f = (f(r_1), ..., f(r_n))$ the final resources array on R and $W = (w_{i,j})_{i,j} \in \mathcal{M}_n(\mathbb{R})$ the weighting matrix obtained by NBI.
Then we have:

$$f = W f_0$$

The matrix coefficient $w_{i,j}$ represents the fraction of resource that r_j sends to r_i.

$$w_{i,j} = \frac{1}{\deg(r_j)} \sum_{k=1}^{m} \frac{a_{i,k} a_{j,k}}{\deg(u_k)}$$

with

$$a_{i,k} = \begin{cases} 1 & \text{if } (r_i, u_k) \in E \\ 0 & \text{otherwise} \end{cases}$$

4.5.3 Application to recommendation. We can now apply NBI to build a recommendation system.

In [9], Zhou et al. used NBI in a slighlty different context. Users were not rating objects but picking them, like buyers would do on an eCommerce website.

We will slightly transform the original "rating" problem to a "picking" problem by saying that a Yelp user picks up a restaurant iif he rated it with a score ≥ 3. Thus the problem becomes binary.

With the same notations, R represents the restaurants while U represents the Yelp users.

Our goal is to predict whether or not user u_k will pick up restaurant r_i, given that he already rated restaurants $r_{i_1}, ..., r_{i_l}$ with scores $\rho_{i_1}, ..., \rho_{i_l}$.

To do so, we project G on the restaurants node set and weight the projected graph with the NBI algorithm. Then we take the initial resource array of user u_k as

$$f_0(u_k) = \begin{bmatrix} 0 \\ \mathbf{1}_{\rho_{i_1} \geq 3} \\ ... \\ \mathbf{1}_{\rho_{i_l} \geq 3} \\ 0 \end{bmatrix}$$

Each user has his own initial resource array.

The final resources $\pi_1, ..., \pi_n \in \mathbb{R}^n$ are deduced from the final resource array:

$$f(u_k) = W f_0(u_k) = \begin{bmatrix} \pi_1 \\ \pi_2 \\ ... \\ \pi_{n-1} \\ \pi_n \end{bmatrix}$$

The resources are then sorted in descending order:
$\pi_1^* \geq \pi_2^* \geq ... \geq \pi_{n-1}^* \geq \pi_n^*$.

We make the assumption that the ratings follow a uniform distribution over $\{1; 5\}$. Therefore the first 60% of the sorted resources are predicted pickups.

4.5.4 Results. We solved a binary problem therefore we will rather use Accuracy as a metric rather than the Mean Squarred Error (MSE) or the Mean Absolute Error (MAE).

$$\text{Accuracy} = \frac{\text{True positive}}{\text{True positive + False positive}}$$

As before, we used only the data from Illinois as the entire Yelp dataset was too large. We achieve 71% accuracy on the test set, which contains 3000 new ratings. Because the performance of this method is not measurable thanks to MAE and RMSE, it will not be compared to the other methods.

4.6 Rating Prediction Based on User-User Unipartite Graph: User-based Collaborative Filtering

From the graph representing friendships between users, it is possible to build a new similarity matrix between users.
There is a variety of similarity measures for analyzing the proximity of nodes between users. We have been working on two of them: the Jaccard Coefficient and the FriendTNS.

For two users u and v of U, these two similarity measures are:

$$sim_{Jaccard}(u,v) = \frac{|\Gamma(u) \cap \Gamma(v)|}{|\Gamma(u) \cup \Gamma(v)|}$$

where $(\Gamma(u))$ denotes the set of neighbors of u.

$$sim_{FriendTNS}(u,v) = \begin{cases} 0 & \text{if there is no edge between u and v} \\ \frac{1}{deg(u)+deg(v)-1} & \text{otherwise.} \end{cases}$$

where $deg(u)$ denotes the degree of u.

Then, the rating prediction formula is the same as earlier for the user-based collaborative filtering method.

$$\hat{r}_{u,i} = \frac{\sum_{\forall v \in U}(sim(u,v) r_{v,i})}{\sum_{\forall v \in V} sim(u,v)}$$

For « unknown » users, for which the similarities with other users are all zero, the calculation leads to a division by zero. In this case, the predicted rating is set to the average rating of the train set.

4.7 Rating Prediction Based on Multi Graph: User-based Collaborative Filtering

It is possible to build another user-based collaborative filtering method from the two presented above. This method combines the two similarity matrices obtained for the bipartite users-restaurants graph and the unipartite users graph in the sections 4.1 and 4.6. These matrices are respectively noted $(sim_B(u,v))_{(u,v)\in U^2}$ and $(sim_A(u,v))_{(u,v)\in U^2}$.

In this new method, the similarities between users are stored in a new similarity matrix computed from the other two.
Firstly, in several cases the distribution of the similarity values in the interval [0,1] between the two matrices differ, thus, it is unfair to take a simple weighted average of them. In order to solve this problem, we first apply the following transformation to the matrices A and B:

For X = A or B, $sim_X(u,v) = \frac{sim_X(u,v) - m_X}{s_X}$ with m_X the mean similarity value of the matrix X and s_X the standard deviation value of the matrix X.
We finally scale the obtained similarity values back in the interval [0,1].

Then, we compute the new similarity matrix thanks to the following formula:

$$sim(u,v) = a(u) \cdot sim_A(u,v) + (1 - a(u)) \cdot sim_B(u,v)$$

Where:

$$a(u) = \frac{dA(u)}{dA(u) + dB(u)}$$

Where:

- $dA(u) = \frac{n \cdot deg_{uni}(u)}{\text{number of non zero values in the adjacency matrix of the users graph}}$, with n the number of users and $deg_{uni}(u)$ the number of friends of the user u, represents the local to global density of the selected user into the adjacency matrix of the users graph.
- $dB(u) = \frac{n \cdot deg_{bi}(u)}{\text{number of non zero values in the rating matrix of the users-rest graph}}$, with n the number of users and $deg_{bi}(u)$ the number of ratings given by the user u, represents the local to global density of the selected user into the rating matrix of the users-restaurants graph.

Thanks to this new similarity matrix, we can compute the ratings predictions thanks to the same formula as earlier.

$$\hat{r}_{u,i} = \frac{\sum_{\forall v \in U}(sim(u,v) r_{v,i})}{\sum_{\forall v \in V} sim(u,v)}$$

For « unknown » users, for which the similarities with other users are all zero, the calculation leads to a division by zero. In this case, the predicted rating is set to the average rating of the train set.

We also tried a variant of this method, with predictions given by the following formula:

$$\hat{r}_{u,i} = avg_u + \frac{\sum_{\forall v \in U}(sim(u,v) \cdot \mid r_{v,i} - avg_v \mid)}{\sum_{\forall v \in V} sim(u,v)}$$

Where avg_u is the average known ratings of the user u, or the average rating of the train set for unknown users.

4.8 Rating Prediction by Regression

A very different approach to rating prediction is to consider it as a classical regression problem. Even if this method is outside the scope of the class, we thought it would be relevant to compare its performance with graph-based methods.
The first step is to restructure the data: we need a ratings vector R and a features matrix X. In order to build the features matrix, we keep the relevant features of the business and the user associated to each rating. A big difference with the other methods is that we drop all the information regarding friendships. After some basic feature engineering we get a features matrix with 19 columns.

The second step is to choose and train a relevant regression model. Random forest regression has been chosen because it is adapted to the non-linear relationships in the data set and performs well when evaluated with cross-validation on the training set. The value of the

main hyper-parameters is determined with a grid-search. Finally, the tuned model is trained on the full training set.

5 EVALUATION

5.1 Dataset

The dataset used for the evaluation of the methods presented previously was presented earlier in this report.

5.2 Evaluation metrics

In order to measure the performance of our models thanks to the results obtained on the test set of selected dataset, regression metrics were used.
In this way, we used the Root Mean Square Error (RMSE) and the Mean Absolute Error (MAE).

The RMSE is a quadratic scoring that measures the average magnitude of the errors in a set of predictions, without considering their direction. It is the square root of the average of squared differences between prediction and actual observation.

$$RMSE = \sqrt{\frac{1}{n}\sum_{j=1}^{n}(y_j - \hat{y_j})^2}$$

The MAE also measures the average magnitude of the error. It is the average over the test sample of the absolute differences between prediction and actual observation.

$$MAE = \frac{1}{n}\sum_{j=1}^{n}|y_j - \hat{y_j}|$$

Thus, these metrics tell us how close predicted ratings are to actual ratings.

5.3 Results obtained

In order to have a reference value, we first computed the values of the RMSE and the MAE when we assign the average rating of the train set to every couple (user, restaurant) of the test set. We obtained:

- RMSE = 1.40
- MAE = 1.21

For our different models, we obtained the following results:

Method	RMSE	MAE
Bipartite User CF	1.37	1.10
Bipartite Item CF	1.49	1.12
SVD Matrix Factorization	1.26	1.04
SVD++ Matrix Factorization	1.57	1.03
Deep Matrix Factorization	1.23	0.97
Unipartite User CF (Jaccard)	1.41	1.19
Unipartite User CF (FriendTNS)	1.40	1.18
Multi-Graph User CF	1.36	1.09
Multi-Graph User CF Variation	1.69	1.29
Random Forest Regression	1.14	0.90

As explained in the dedicated part, the Network Based Inference method does not adapt well to the metrics used here. In order to evaluate the performance of this algorithm, we used the accuracy, and obtained 71% on the test set.

6 CONCLUSION

We have seen that many methods can be used to solve the problem studied, and can lead to interesting results.

The results obtained show that the usual user-based Collaborative Filtering outperform item-based Collaborative Filtering. This can be explained by the fact that the number of restaurants studied is lower than the number of users, but this configuration is common in the real world, we can therefore say that user-based Collaborative Filtering is usually better to use than item-based Collaborative Filtering.
In addition, we can notice that the addition of information extracted from the explicit friendships social network of the Yelp website allows to improve the performance of Collaborative Filtering.
Moreover, Latent Collaborative Filtering methods allow to achieve better results than other Collaborative Filtering methods, especially when Deep Learning is used. Thus, the Deep Matrix Factorization method is the best graphical based method.

However, the best results are obtained using a method that does not involve networks and graphs analysis: Regression.

It is important to note that we worked on a small dataset due to the limited time available. With more time, we could have built our models on larger datasets and surely achieve better results.

The main idea for future work is to study how to combine the best studied methods to build a better recommendation system. For exemple, adding graphical features in the regression model is a possibility, or building a general model by stacking the others would be interesting.

REFERENCES

[1] Yelp open dataset. *Yelp [online], Available at https://www.yelp.com/dataset*, 2020.
[2] H.-J. X. andX. Y. Dai, J. Zhang, S. Huang, and J. Chen. Deep matrix factorization models for recommender systems. *Twenty-Six International Joint Conference on Artificial Intelligence*, pages 3203–3209, 2017.
[3] S. Funk. Netflix update: Try this at home. *The Evolution of Cybernetics [online], Available at https://sifter.org/~simon/journal/20061211.html*, 2006.
[4] J. HERLOCKER, J. KONSTAN, L. TERVEEN, and J. RIEDL. Evaluating collaborative filtering recommender systems. *ACM Transactions on Information Systems*, 22(1):5–53, 2004.
[5] Y. Koren, R.Bell, and C.Volinsky. Matrix factorization techniques for recommender systems. *Datajobs.com [online], Available at https://datajobs.com/data-science-repo/Recommender-Systems-[Netflix].pdf*, 2009.
[6] V. Lazovskiy. Predicting film ratings with simple linear regression. *Towards Data Science [online], Available at https://towardsdatascience.com/predicting-film-ratings-with-simple-linear-regression-cabe35bddcde*, 2018.
[7] J. Schafer, D. Frankowski, J. Herlocker, and S. Sen. Collaborative filtering recommender systems. *The Adaptative Web, LNCS 4321 (P. Brusilovsky, A. Kobsa, and W. Nejdl)*, pages 291–324, 2007.
[8] P. Symeonidis, E. Tiakas, and Y. Manolopoulos. Product recommendation and rating prediction based on multi-modal social networks. *RecSys '11*, 2011.
[9] T. Zhou, J. Ren, M. Medo, and Y.-C. Zhang. Bipartite network projection and personal recommendation. *Physical review E*, 76(4):046115, 2007.