# NYC Restaurant Recommendation system

**Final Project Report**

Lucas DE SOUZA BALANCIN
lucas.balancin@student-cs.fr
ESSEC & CentraleSupélec
Paris, France

Olesia KHRAPUNOVA
olesia.khrapunova@student-cs.fr
ESSEC & CentraleSupélec
Paris, France

Valentina JERUSALMI
valentina.jerusalmi@student-cs.fr
ESSEC & CentraleSupélec
Paris, France

Rodolphe ROYER DE LA BASTIE
rodolphe.delabastie@student-cs.fr
ESSEC & CentraleSupélec
Paris, France

## KEYWORDS

recommendation system, restaurants, sentiment analysis

## 1 ABSTRACT

In this paper we present the results of our final project in Machine Learning in Network Science. We will describe the development and evaluation of several bipartite graph-based approaches that can recommend new restaurants to users by using the user's historical data. Our work is based on the New York City Restaurant Rich Dataset from Foursquare, a geolocation social media. A key challenge was building a robust recommendation system using a small amount of data.

## 2 MOTIVATION

Imagine the following situation: you plan on spending a few days in a city that you have never visited before. At some point, you will go out and eat in a restaurant. The question is: how will you select this restaurant? Now let's say that we are a few years ago when the internet was not as widely used as today. If you are in luck, you might have a relative or an acquaintance that knows restaurants in this city. Hence you ask this acquaintance for a recommendation of which venues I am likely to enjoy. This acquaintance will, consciously or subconsciously, try to remember from your past experiences spent together all the aspects of a restaurant that you might enjoy and make you a small list of venues that you should try. This recommendation is reassuring and is a valuable insight as you now go to a venue that was recommended by someone you trust, and it will likely improve the likelihood that you enjoy this venue as opposed to eating in a randomly selected venue.

Now back in 2021, we do have the internet that is widely used. According to BrightLocal[1], 93% of consumers used the internet or related platforms to search for local business in 2020, and restaurants were the favorite industry category. Additionally, OpenTable[2] reports that 90% of people eating at restaurants search for a restaurant online before deciding where to eat. Clearly, the internet has become a tool to search for and help you decide where to eat. An internet platform that has gathered over time data of where you have eaten and weather you enjoyed or disliked each venue can be used to make new recommendations and improve your restaurant experiences. In a sense, an automatic recommendation system can scale up this human process of asking an acquaintances for recommendations.

[1] https://www.brightlocal.com/research/local-consumer-review-survey/
[2] https://restaurant.opentable.com/news/tips/making-the-grade-how-do-your-restaurants-digital-marketing-efforts-stack-up/

*Machine Learning in Network Science,*
.

## 3 PROBLEM DEFINITION

The objective of this project is to construct a recommendation system based on past behaviour across different users. We wish to recommend a set of previously unvisited venues that the user is likely to enjoy for each user. The problem to solve is to predict if a specific user would like a venue given his past activities and characteristics of the venue. We plan to solve this using the experience across all the users and similarities between users and venues to compute these recommendations. One fundamental assumption throughout our report is that users who go to the same places are likely to have similar tastes and preferences in restaurants.

For this task, we will use a bipartite graph. In graph theory, this is a type of graph that has two sets of distinct nodes, called parts. Edges can only connect nodes from different parts. In other words, there are no edges among the nodes of the same set.

Formally we will have:

- The bipartite graph $G = (U, V, E)$
- The set of users is $U$
- The set of venues is $V$
- The set of edges linking one user to a venue is $E$, each link is $e_{ij}$ with $i$ the user id and $j$ the venue id

## 4 RELATED WORK

The applications of recommendation systems are broad and can serve multi-purposes given the same approach. We are in the case of a bipartite recommendation. This means that, based on one entity (users), we wish to recommend a different entity (venues). This is opposed to recommendation systems where the two entities are of the same type — for example, recommending to a user (users) profiles to follow on a social network (users). We can note that link prediction on bipartite graphs is a less researched topic than the single node class links.

There are several related works done on bipartite recommendation systems that extensively use collected data about their users.

- Product recommendation to customers on Amazon
- Music recommendation to users on Spotify

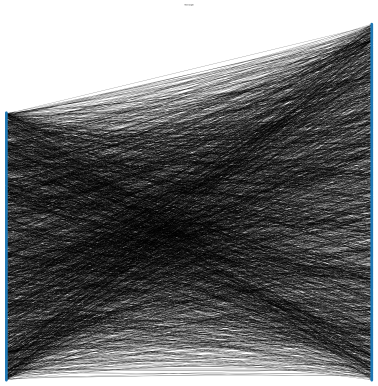

**Figure 1: Graph structure**

- Movie recommendation to viewers on Netflix

Recommendation systems can have a crucial economic value. It has been reported by Gomez-Uribe and Hunt (2015) [1] that 80% of choices to view a video on Netflix are attributed to the recommendation system. In the now famous 2009 Netflix prize, the winning team of Y. Koren, R. Bell and C. Volinsky [5] used a matrix factorization technique to recommend movies to users. This method decomposes the user-movie interaction matrix and learns the latent factors that describe this interaction. This approach shows great results but does not take full advantage of the network structure.

A group of Stanford students [3] tried predicting a user reviewing a business using the Yelp Challenge dataset. We adapted some of their ideas (notably, similairty measures applied to bipartite graphs) for our link prediction algorithms.

# 5 NETWORK

## 5.1 Data

To address the objectives of this project, the dataset "NYC Restaurant Rich Dataset" [4] will be used. This dataset includes check-in, comments and tag data of restaurant venues in NYC collected from Foursquare from 24 October 2011 until 20 February 2012. The data is split into three tables:

- check-ins.csv - Contains the list of check ins (a user_id - venue_id pair)
- tags.csv - Contains a list of tags describing a venue, for each venue
- comments.csv - Contains a list of comments left by users to venues

By looking at the characteristics of these datasets we have:

- 3 112 unique users
- 3 298 unique venues
- 27 149 check-ins: a user checking in a venue (user can have multiple check-ins to the same venue)
- 10 377 comments: left by users for venues
- 2 602 lists of tags: given by users, in aggregate, to venues (not each venue had tags)

[3] http://snap.stanford.edu/class/cs224w-2014/projects2014/cs224w-82-final.pdf
[4] https://sites.google.com/site/yangdingqi/home/foursquare-dataset

## 5.2 Network Construction

We decided to construct our network using the NetworkX python library as this is the one we used during the in-class exercises and it fits perfectly to what we wish to accomplish.

The first step was to add the nodes of the network. As this is a bipartite graph, we have two types of nodes: users and venues. Each unique user and unique venue become a node of the network. Our next step was to add the edges between users and venues. There are several observation wroth mentioning for this step.

- We started by creating an edge for each check in pair linking a user to a venue.
- We observed that there are many duplicate check in data. This means that a user went more than once to a venue. We include this information which is relevant to determine a user preference by adding an attribute the each edge corresponding to the number of visits to the corresponding venue.
- Some comments were left by users to a venue but this user-venue pair did not appear in the check in data. We assumed that these users left a comment after having visited a venue but might not have manually checked in. Hence the user-venue pairs in the comment table are considered as a visit and are added as edges to the network.

Once the network construction was completed, we took a closer look at some key characteristics of the graph. This revealed the following details about network structure:

- Number of connected components: 109
- Fraction of nodes in GCC: 0.96
- Fraction of edges in GCC: 0.993
- Average degree: 6.69

## 5.3 Sentiment Analysis

Our first step before making new recommendations was to assign weights to existing edges, which would indicate the level of affinity of a user towards a venue. To determine these weights we used two sets of information:

(1) Some edges had an associated review. When this was the case, we performed a sentiment analysis on the comment.
(2) We also took into account the number of times this user went to this venue. The underlying assumption was that a user who went many times to the same venue most certainly appreciated it very much i.e. "actions speak louder than words".

To perform sentiment analysis on the reviews, we used a transformers library from a Hugging Face[5] and a pre-trained model from NLPTown [6] using BERT for product reviews. Two examples can be seen in Figure 2.

Since we had probabilities of belonging to each class of the stars, we decided to use them as weights to average the classes and get the final score. Since we know reviews can contain ironies and mixed feelings, this strategy may help to have a closer score to the real truth.

[5] https://huggingface.co/
[6] https://huggingface.co/nlptown/bert-base-multilingual-uncased-sentiment


**Figure 2: Sentiment classification in two different sentences**

At the end of this process, we also took into account the number of times a person went to a venue. Since we didn't have sentiment for each visit, we set a sentiment classification for each edge, following the procedure:

- If a pair user-venue had more reviews than check-ins, the weight of the edges would be the sum of all the reviews. For example, if a user reviewed the same venue twice with the score of 4 stars, the edge would have weight **8**.
- If we had more check-ins, this meant that the user didn't mind reviewing all his visits, so we classified that as a neutral behavior. Therefore, these unreviewed check-ins had weight as the *average* of the user for other venues.
- In case that the user didn't have evaluations to consider as average, the visit got the average of the venue.
- In the last case, if both user and venue didn't have evaluations, the visit was set as **3** stars.

So, at the end, we had each edge weighted as the following example:

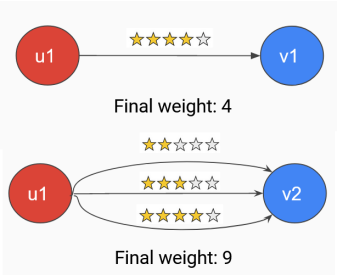


**Figure 3: Example of the weights of the edges**

For our recommendation system, we wished to only take into account positively classified edges. Hence for the two recommendation system types that we developed and implemented we only kept the edges with a rating of 3 stars or more. This implied that the user didn't go more than once with reviews below 3. The discarded edges would later serve the purpose of scoring custom recommendation system. The final distribution of the edges weight is shown in the Figure 4.

After removing the edges with a rating below 3, we recalculated graph characteristics and got the following results:

- Number of connected components: 248
- Fraction of nodes in GCC: 0.937
- Fraction of edges in GCC: 0.992

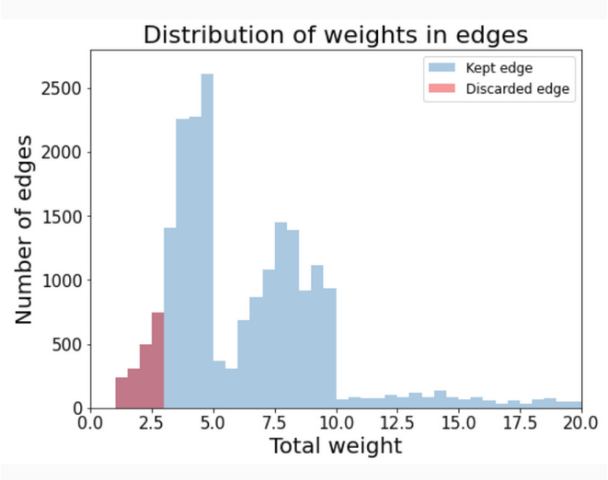


**Figure 4: Distribution of the final weight of the edges**

- Average degree: 6.18

## 6 RECOMMENDATION SYSTEM 1: LINK PREDICTION APPROACH

For our recommender system, we first focused on link prediction approaches. As the only edges included in the final graph indicated positive prior experiences, the predicted links would suggest venues the user is likely to visit and enjoy (rather than just visit and potentially dislike). We pursued both supervised and unsupervised link prediction strategies, as described in the following subsections.

The final constructed graph was not connected. Therefore, we decided to focus only on the largest connected component for these approaches, as it was desirable for our algorithms to work with a fully connected graph. As GCC contained almost all the information about the network (94% of nodes and 99% of edges), we felt comfortable focusing on it for our recommendation task. Moreover, the nodes and edges not included in GCC indicated connections between people who seemed to rarely go out and, if they did, went to places no one else liked. These people and venues would be of a lesser interest for a recommendation system.

In addition, it is worth noting that although we had weight information available for all existing edges, we treated the graph as unweighted for the link prediction algorithms. The main purpose of weights in this case was to discard edges indicating negative experiences, as previously described.

### 6.1 Unsupervised link prediction

We based unsupervised link prediction algorithm on three measures: Common Neighbors, Jaccard's coefficient, and Adamic/Adar. These metrics are often used for link prediction tasks, as they compute similarities between two nodes based on how much their neighbors overlap. The assumption is that if two neighboring nodes have similar neighbors, they should be connected (for example, if two people in a social network share the same friends, they are probably friends as well). However, we could not directly apply these similarity measures to a bipartite graph. If we want to predict

an edge between a user and a venue and look at their common neighbors, we would see that there are no common neighbors, as the user's neighbors are only venues and the venue's neighbors are only users. Therefore, we had to adapt the similarity measures for the new scenario. Specifically, instead of comparing sets of direct neighbors for both nodes, we compared a set of one-hop neighbors ($\Gamma$) for one node and a set of two-hop neighbors ($H$) for the other node. To evaluate similarity between $x$ and $y$ coming from different sides of a bipartite graph, we looked at the closeness between set $\Gamma(x)$ and set $H(y)$. This ensured that sets $\Gamma(x)$ and $H(y)$ included nodes of the same type, even though $x$ and $y$ were of different types. This approach makes an intuitive sense: if many of the users who liked same venues as you (your two-hop neighbors) also like venue A (the venue's one-hop neighbors), you are likely to enjoy venue A too. The adopted measures were calculated as follows:

- Common Neighbors: $|\Gamma(x) \cap H(y)|$
- Jaccard's Coefficient: $\frac{|\Gamma(x) \cap H(y)|}{|\Gamma(x) \cup H(y)|}$
- Adamic/Adar: $\sum_{z \in \Gamma(x) \cap H(y)} \frac{1}{log|\Gamma(z)|}$

Each of these measures could be implemented in two ways: by taking one-hop neighbors of user and two-hop neighbors of venue (marked U-V) and vice versa (one-hop for venue, two-hop for user, marker V-U). All the configurations were tested and compared.

## 6.2 Supervised link prediction

For supervised link prediction, we trained several models to perform binary classification given a user-venue pair. If a model predicted label 1 (link exists), we would recommend the venue to the user. If model predicted label 0 (link does not exist), we would not recommend this restaurant to the person. For this approach, we used all the similarity measures developed for unsupervised link prediction and added several features based on graph structure and node attributes. Specifically, the final feature vector included (assumptions for adding the features are included for new attributes in italics):

- Common Neighbors (both U-V and V-U)
- Jaccard's coefficient (both U-V and V-U)
- Adamic/Adar index (both U-V and V-U)
- Degree of user - *the more the person goes out, the more likely they would be to try a new venue*
- Degree of venue - *the more popular the venue is, the more likely it will be liked by a new visitor*
- Average sentiment towards the venue - *the more positively people talk of the venue, the more liked it is and the more likely a new visitor would enjoy it*
- Average number of visit to the venue - *the more often people return to the venue, the more liked it is and the more likely a new visitor would enjoy it*
- Venue tags (only top 3 most popular) - *some venue types might be more liked*
- Common tags between the venue and other venues the person visited - *if the venue has similar food/atmosphere to the ones the user enjoyed before, he/she is more likely to go and enjoy the venue*

We trained multiple models using these features: SVM, Logistic Regression, Random Forest, AdaBoost, XGBoost, and Extreme Randomized Classifier.

## 6.3 Evaluation

In order to evaluate the different link prediction approaches, we created a train and a test set of edges (0.7/0.3 ratio). Both were balanced, and included both positive and negative samples. We used the train set to train supervised link prediction models, and then evaluated the models' quality on the test set. Unsupervised approaches were directly evaluated on the test set, as there was no training involved in the algorithms. We used area under the curve (auc) of ROC curve to measure model performance.

Below are results from the different models. For unsupervised approaches, we observed that U-V and V-U similarities produced almost the same auc scores (biggest difference was 0,002 for jaccard), so we will only report best score of the two. For supervised link prediction, only the score for the best model (XGBoost) is reported.

| Model | AUC |
|---|---|
| Common Neighbors (V-U) | 0.648 |
| Jaccard (U-V) | 0.643 |
| Adamic/Adar (U-V) | 0.656 |
| Supervised (XGB) | 0.731 |

In the following graph we report the ROC AUC Curve for our best performing model: XGBoost.

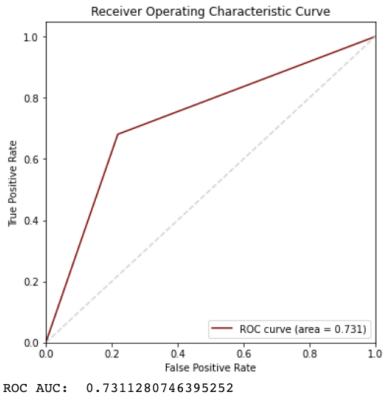


**Figure 5: ROC AUC for XGBoost**

Supervised link prediction approach provided the best results. The additional information provided in the feature vectors must have been useful to make correct predictions. It is worth mentioning that we achieved the best result with only some of the originally included features. Specifically, the most performant model used average number of visits and average sentiment towards the venue, common tags between the venue and previously-visited venues, degree of user and venue, Adamic/Adar (U-V and V-U) and Jaccard's coefficient (U-V). Based on permutation importance, user and venue degree and average number of visits to the venue were the most informative.

# 7 RECOMMENDATION SYSTEM 2: CUSTOM MADE APPROACH

The second approach we tried can be more closely described as an algorithmic method. As the link prediction method, we used only edges included in the original graph that indicate a positive prior experiences. The general idea is the following: for each user, we look at its most preferred venue. We then look at the other users who also went to this venue and count which are their most preferred venue, which in turn becomes our first recommendation.

As a reminder, we know the preferences of an edge (user-venue) thanks to the sentiment analysis.

We constructed two dictionaries:

- A fixed dictionary storing user preferences: Key: user, Value: list of most preferred venues ordered by preferences
- A dictionary that will be computed at every iteration. It stores the favorite venues of all the other users. The venue with the highest occurrence will be our recommendation : Key: venue, Value: occurrences

The following steps describe in details how the recommended venues are computed for each user.

(1) For each user $u_i$, we select its first favorite venue $v_{1j}$.
(2) We select all the users $u_{\setminus i}$ that went and loved this venue $v_{1j}$ (meaning a positive link $e_{\setminus ij}$ exists)
(3) We then select these user's favorite venues and do a majority vote among these venues
(4) If this voted venue is not among the 5 favorite venue of our user $u_{i,}$ then this venue is our first recommendation.
(5) We implement this same reasoning for the user's second favorite, third, fourth and fifth. This process will give us 5 recommendations per users.

We also tried a different approach that works as follows:

(1) For each user $u_i$, we select its first favorite venue (only the first one in this approach) $v_{1j}$.
(2) We select all the users $u_{\setminus i}$ that went and loved this venue $v_{1j}$ (meaning a positive link $e_{\setminus ij}$ exists)
(3) We then select these user's favorite venues and do a weighted vote among these venues based on how much they liked their favorite one. We implement a ranking of all users (this means that starting from the favorite vote of our user of interest, we then select directly all the final recommendations).
(4) If the favorite 5 venues of the other users are not among the 10 favorite venues of our user $u_i$, then this venue is our first recommendation.

## 7.1 Evaluation

To evaluate the quality of our recommendation system we developed a custom-made metric.

Our metric tries to make the the parallel with type 1 errors of hypothesis testing, so the kind of error in which you incur into the false positives. In our context, a false positive correspond to the case in which you recommend a disliked venue, so when you say that something is good but it isn't. In our evaluation, this kind of error is way worse than "forgetting" to recommend a good venue to an user. The evaluation, therefore, proceeds as follows:

To develop this evaluation, we developed the following method:

(1) Pick a user who went to at least 5 venues and liked them, and who went to at least one negative venue he/she disliked.
(2) Launch our recommendation system for this user. This will output a set of n recommendations. In this case we will evaluate on the top 10 ranked venues.
(3) If among the 10 recommended venues, one of them was previously classified as a negative venue by the user, then we consider that the recommendation is a failure and we set the criteria score to 1 for this user. On the other hand, if the recommended venues were not negatively classified by the user, then the second criteria is met and we set its score to 0.

Again, we iteratively compute this binary score for all users were we have enough existing data. We then sum these individual scores and divide by the number of users for which it was evaluated to get the second criteria score. We wish to minimize this score.

Our results are pretty satisfying. On a total number of around 1800 users, we predicted an disliked venue in the first ten recommendations in only 0.5% of the cases for the version 1 and in 1% of the cases for the version 2.

Because of the fact that version 2 is computationally more expensive and apparently slightly less performing than version 1, version 1 is our final choice for the recommendation system.

# 8 LIMITATIONS

The methods that we described assume that each user has already checked in at least a few venues and posted some comments. It does not take into account a new user registering on the platform. The case of recommending venues to a new user is called a cold start recommendation. For this scenario, we cannot use our methodology. One of the options is to start by recommending the top 5 venues overall. Once enough data is collected on this user, our approaches can be applied.

# 9 NEXT STEPS AND BUSINESS VALUE

Another use of our recommendation system is for advertising purposes. Right now we are recommending venues to users. The aim of this recommendation system is to improve the user experience, which will in turn make the user come back to the platform. As the user comes back to add and rate more restaurants the recommendation system will become increasingly accurate, creating a positive feedback loop. This recommendation system could be seen as "organic".

Our "organic" recommendation system could be adapted to be reversed: recommending a list users that are likely to enjoy a venue. It is important to identify the business value of such a recommendation system. For example, a venue could be interested in launching a marketing campaign to attract more customers. To do so, our "reversed" recommendation system could output a list of users that would potentially enjoy going to this venue. This list of users would then serve as the basis of a targeted marketing campaign.

By making the parallel with Google Search, the "organic" recommendation system that we developed can be seen as the organic search results, and the "reversed" recommendation system that gives a list of users interested in a venue can be seen as the paid search results. Of course, this list of users would not be given for free to venues. The most common approach of today's two sided

platforms is to make venues pay for this marketing campaign that would be showed on the platform itself. This is one of the meaning behind the "exploitation of personal data" expression.

Having these two recommendation systems attract both entities of the platform. Users are satisfied because they are given quality recommendations, and venues are satisfied because they can target more efficiently new potential clients.

## 10 CONCLUSIONS

In this report, we showed graph approaches to recommendation system for restaurants. By exploring connections between users and venues and their sentiment regarding each venue, we constructed a graph in which edges indicated only positive experiences. Using this graph, we created link prediction algorithms based on unsupervised and supervised methods, which evaluated if a user would be likely to link (have a positive experience) to unknown restaurants. The supervised approach performed better, with ROC AUC of 0.73. We also managed to construct a custom recommendation algorithm by using an intersection between favorite venues of a set of users. Recommendations made using the custom method didn't include a proposition that the user would dislike in 99.5% of the cases. The two types of approaches described in this paper are rather different and could not be easily compared. However, they provide ideas for different ways recommendation systems could be designed. In a real life scenario, they could be compared by collecting feedback about the quality of recommendations made by the systems (for example, by asking users to rate a recommended venue after going to it) and comparing the results (which approach's recommendations are rated higher, on average).

## REFERENCES

[1] Gomez-Uribe and Hunt. 2015. The Netflix Recommender System: Algorithms, Business Value, and Innovation. *ACM Transactions on Management Information Systems* (2015), Article No.: 13.

[2] Jeroome Kunegis, Ernesto W. De Luca, and Sahin Albayrak. 2010. The link prediction problem in bipartite networks. *Proceedings of the Computational intelligence for knowledge-based systems design, and 13th international conference on Information processing and management of uncertainty* (2010), 380–389.

[3] T. Lakshmi and S. Bhavani. 2021. Link prediction approach to recommender systems. (2021).

[4] Jing Li, Lingling Zhang, Fan Meng, and Fenhua Li. 2014. Recommendation algorithm based on link prediction and domain knowledge in retail transactions. *Procedia Computer Science* 31 (2014), 875–881.

[5] R. Bell Y. Koren and C. Volinsky. 2009. Matrix Factorization Techniques for Recommender Systems. *Computer* (2009), vol. 42, no. 8, pp. 30–37.

[6] Dingqi Yang, Daqing Zhang, Zhiyong Yu, and Zhu Wang. 2013. A sentiment-enhanced personalized location recommendation system. (2013), 119–128.

[7] Guangchao Yuan, Pradeep K. Murukannaiah, and Zhe Zhang. 2014. Exploiting sentiment homophily for link prediction. *8th ACM Conference on Recommender Systems* (2014), 17–24.