Open Access Library Journal
2023, Volume 10, e10009
ISSN Online: 2333-9721
ISSN Print: 2333-9705

Design and Implementation of a Recommender System for Tourist Visit Management

Christophe Lwanyi Ashimalu, Simon Ntumba Badibanga, Pierre Kafunda Katalay

Department of Mathematics, Statistics and Computer Science, University of Kinshasa, Kinshasa, Democratic Republic of the Congo
Email: abbechristophelwanyi@yahoo.fr

How to cite this paper: Lwanyi Ashimalu, C., Ntumba Badibanga, S. and Kafunda Katalay, P. (2023) Design and Implementation of a Recommender System for Tourist Visit Management. *Open Access Library Journal*, **10**: e10009.
https://doi.org/10.4236/oalib.1110009

Received: March 14, 2023
Accepted: May 28, 2023
Published: May 31, 2023

Abstract

Recommender systems are currently applied in many fields. They try to provide users with recommendation services based on their personalized preferences to reduce the ever increasing amount of information online. With the number of mobile phone users growing exponentially, travel guides have become an increasingly important search tool in recent years. Of course, we are not and will not be the first to implement a prototype to offer recommendations to users (tourists). Our particularity and/or novelty in this paper is to present a recommendation system to capture the optimal route taking into account both cost and distance constraints. The open dataset used covers information on tourist trip reviews of thousands of tourists who have visited different attractions in Italy and around the world. An association rule based on the exploratory approach will take into account the contextual information of the user's actual location to produce a dataset to be followed. In addition, a case study on tourism, Tourist visits, is implemented to verify the feasibility and applicability of the proposed system. The results of this work indicate that the proposed system has great potential to prepare the planning of tourists based on the use of mobile phones.

Subject Areas

Artificial Intelligence

Keywords

System, Recommendation, Constraints, Association Rules, Exploratory Approach

1. Introduction

This study is structured around four main points. First, a historical overview of recommender systems will be given. This is followed by an analysis of the engineering behind the operation of a recommender system. Next, the notions of association rules and the ROC curve will be discussed. Finally, the analysis of the *InfTech_Tourism*[1] data will allow us to implement a prototype recommendation system for the management of tourist visits. As can be seen, we will follow the analytical-expositive method.

2. Development: Recommendation Systems

Every day of our lives we are confronted with choices to be made, even without wanting to: What to wear? Which product to buy? What clothes to wear? Where to travel on holiday? ... Even though recommender systems are applied in different areas of life, they share the same objectives: to help users make useful choices.

The scope of these decision areas is very wide. The possibilities they can offer are numerous. Evaluating them to find what is best for them is a very difficult and delicate task, and can be very time-consuming.

According to Idir BENOUARET, the ability of computers to make recommendations to users was quickly recognized in the history of computing ([1], p. 9). Quoting Grandy RICH, the author continues “the first step towards recommendation systems was precisely to develop an automatic library management system in the late 1970s” ([1], p. 9).

Although this work was the first serious and interesting attempt to implement a recommender system, its use remained very limited. Its weakness was that it classified users into stereotypes on the basis of a brief interview and used these stereotypes to produce recommendations about books. This gave rise to what would later be called information overload.

So, in order to solve this famous problem of information overload, collaborative filtering was born around the 1990s. Nowadays, there are several types of recommender systems. Recommendation systems are classified according to the approach used to estimate the missing scores.

2.1. Content-Based Filtering

The user will be recommended items that are similar (in the sense of a similarity measure between items) to those he/she has preferred in the past. In order to better understand this method, it is necessary to analyze it in a tripartite approach, the quintessence of which is as follows:

- Based on the objects already evaluated and/or selected: Item-Item
- Based on the user’s profile: User-Item
- Based on the use of a model

2.1.1. The User Profile

- Important criteria

[1]InfTech_Tourism is a file containing data collected by a group of researchers from the city of Lucca, in the Tuscany region of Italy, who have granted us permission to exploit this data.

- Consideration of
  => Boolean comparisons
  => Model generation

2.1.2. Disadvantages

- Requires descriptive content, difficult for films
- Lack of serendipity[2]
- Easily misses interesting recommendations

2.1.3. Developments

Figure 1 shows the content-based system.

- Use of the Semantic Web
- Data description: XML, RDF [2]

2.2. Collaborative Filtering Method [2]-[7]

The user will be recommended items that other users with similar tastes and preferences (in the sense of similarity between users and items) have liked in the past.

- Based on users who are considered similar
- Determination of user groups [Aggregative method and Centralized method]
- Content independent, human factor (aesthetics)
- Any form of content can be involved as long as a human can appreciate it
- Rating matrix

Collaborative filtering is a type of recommendation engine that uses both user data and item data, specifically, individual users' ratings of individual items. In this way, items are recommended based on the ratings of other users, thus collaborative. This data can be represented in a utility matrix, with one axis being the users and one axis being the articles. The aim of collaborative filtering recommendation engines is to fill in the gaps in a utility matrix, as not all users have rated each item, and then to produce the highest rated and previously unranked items as recommendations. (Figure 2)

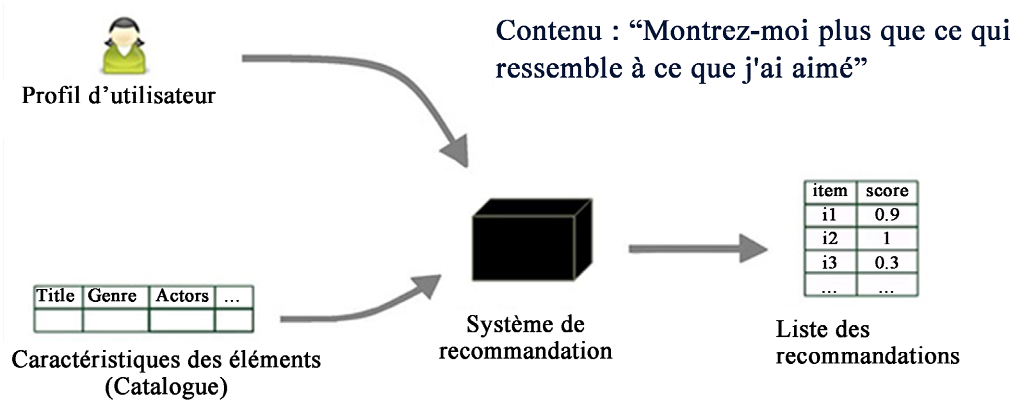

Profil d'utilisateur
Contenu : "Montrez-moi plus que ce qui ressemble à ce que j'ai aimé"
Title | Genre | Actors | ...
Caractéristiques des éléments (Catalogue)
Système de recommandation
item | score
i1 | 0.9
i2 | 1
i3 | 0.3
... | ...
Liste des recommandations

Figure 1. Content-based system

[2]Serendipity is a form of intellectual readiness, which makes it possible to draw rich lessons from an unexpected find or a mistake.

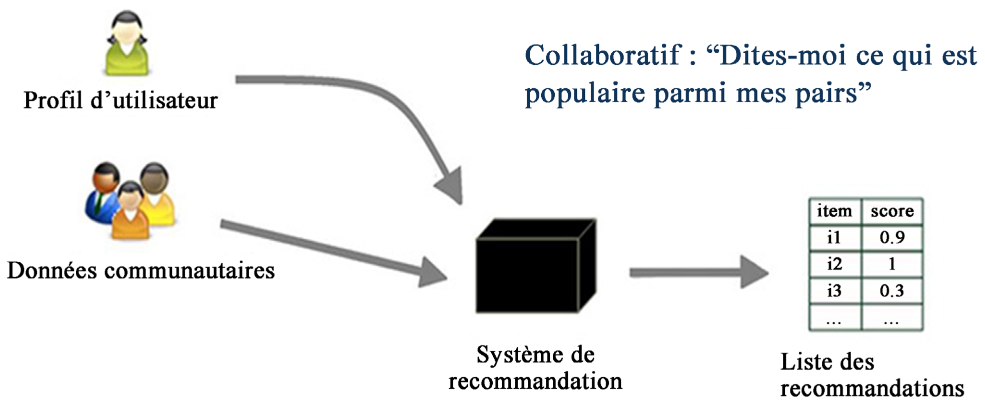

Profil d'utilisateur
Données communautaires
Collaboratif : "Dites-moi ce qui est populaire parmi mes pairs"
Système de recommandation
item score
i1 0.9
i2 1
i3 0.3
... ...
Liste des recommandations

Figure 2. Collaborative system.

2.2.1. Memory-Based Algorithms

- Establish a vote prediction for the user
- Use the average of votes for a user
- Define similarity between users: Pearson correlation, vector similarity...

2.2.2. Problems Related to Filling the Matrix

- Problem of the first vote
- Scattered votes
- Requires many votes for relevance

2.2.3. Scoring Matrix

There are three main techniques for populating the utility matrix using the collaborative filtering method: User-User, Item-Item and Singular Value Decomposition (SVD). We will go through each of these using our simple utility matrix above to try and predict what User 1 would rate Item 3. (Figure 3)

2.2.4. User to User

There are two main steps in calculating the missing value of our utility matrix:

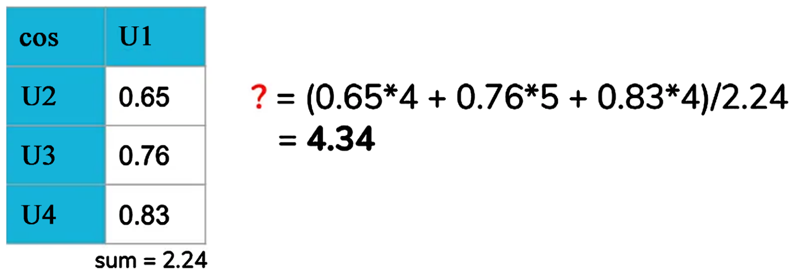
- Calculate the similarity between U1 and all other users
- Calculate U1's score for I3 by taking an average of the other users' I3 scores, weighting each user's score by the user's similarity[3] to U1.

	U1	U2	U3	U4
I1	4	2	3	5
I2	3	2	4	2
I3	?	4	5	4
I4	3	2	4	4

Figure 3. Collaborative filtering matrix.

[3]There are several metrics for calculating similarity [Jaccard, Euclidean distance, Manhattan distance, Minkowski distance, Hamming distance...]. Note that these metrics do not necessarily give the same result. The choice of a metric depends on the result that one would like to achieve.

Example (Figure 4):



cos	U1
U2	0.65
U3	0.76
U4	0.83

sum = 2.24

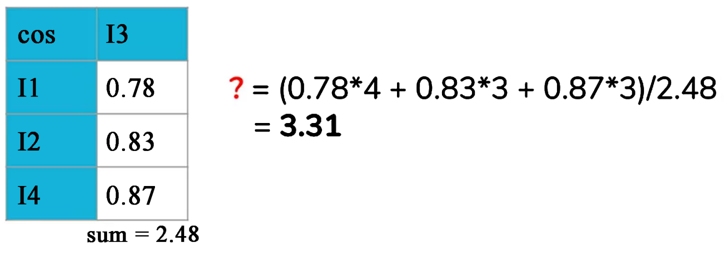
? = (0.65*4 + 0.76*5 + 0.83*4)/2.24
= **4.34**

Figure 4. Left: cosine similarity of U1 with all other users; Right: weighted average scores for I3.

2.2.5. Item to Item

Item to Item collaborative filtering is much the same as User to User, but instead of calculating the similarity between users, it is calculated between Items. The final value calculated is then an average of the other scores in U1, weighted by the similarity of I3 to other items.

Example (Figure 5):



cos	I3
I1	0.78
I2	0.83
I4	0.87

sum = 2.48

? = (0.78*4 + 0.83*3 + 0.87*3)/2.48
= **3.31**

Figure 5. Left: cosine similarity of I3 with all other elements; Right: weighted average of scores for U1.

Adopting the Item to Item approach, we obtain a prediction value of 3.31—very different from the previous value of 4.34. However, a few observations are worth highlighting:

- When calculating similarity, some sources ask to treat missing values as 0 while others simply omit the entire row/column with the missing value in the similarity calculation.
- In general, the Item to Item approaches have been more effective due to the unique tastes of users.
- When deciding whether to use the User to User or Item to Item approach, it is advisable to consider the complexity of the algorithm. If we have m Users and n Items, the time complexity would be O (m^2n) for User to User and O (mn^2) for Item to Item. We can decide to choose Item to Item if and only if we have more users and vice versa.

2.2.6. Singular Value Decomposition (SVD)

Let's start with some theory to understand where these concepts come from. Singular value decomposition (SVD) is a form of matrix factorization. Matrix factorization decomposes a matrix into a product of (usually three) matrices. In algebra, when we factor quadratic equations into their linear parts (*i.e.* $x^2 + 2x + 1 = (x + 1)(x + 1)$), it is a similar idea.

SVD is the model made famous by Simon Funk in the Netflix Prize competition in 2007. Singular value decomposition is usually performed by using the eigenvalues and eigenvectors of a matrix to decompose it into three-component matrices. The name of the algorithm used in the Python libraries to solve recommendation engines is called SVD, but it does not exactly factor the utility matrix. Instead, it does something of the reverse of SVD and tries to recreate the utility matrix using not three, but two component matrices. These two matrices, as illustrated below (**Figure 6**), can be interpreted as the Item matrix and the User matrix.

Example:

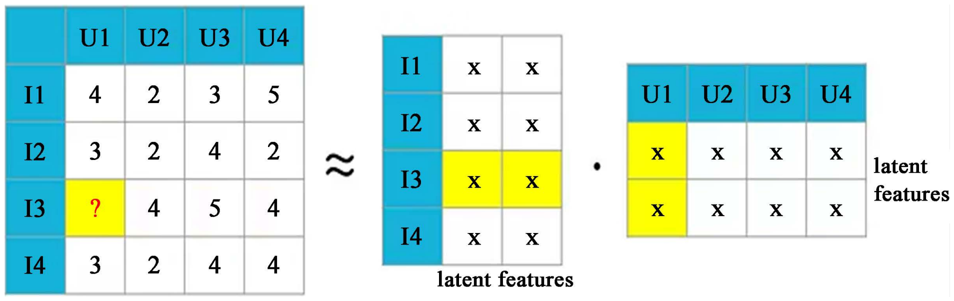


Figure 6. Decomposition of the utility matrix into an item matrix and a user matrix.

The latent features simply refer to an abstraction of all the features of the Items or Users. As long as the same number of latent entities is available for the Items and Users, the matrices can be multiplied to produce a single matrix with the same dimensions as the utility matrix. The number of latent entities is a hyperparameter that can be set in the model. Based on the matrix multiplication, one can also see that the evaluation value of U1 for I3 is affected by the row I3 of the Items matrix and the column U1 of the Users matrix.

Because we cannot decompose the matrices with missing values, we have to take another approach. This is where machine learning comes in. Now we need to recreate the utility matrix with our Items matrix and our Users matrix. This is done using the gradient descent method known as Alternating Least Squares (ALS). (**Figure 7**)

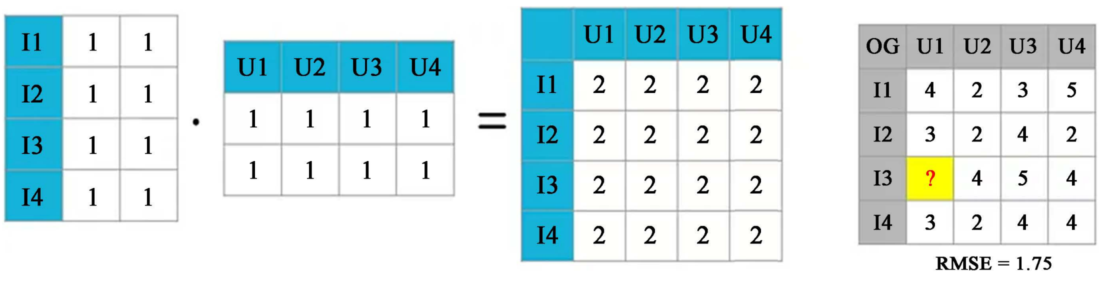


Figure 7. In blue: after initializing the component matrices with 1, the recreated utility matrix is composed of 2. In grey: the original utility matrix for comparison purposes.

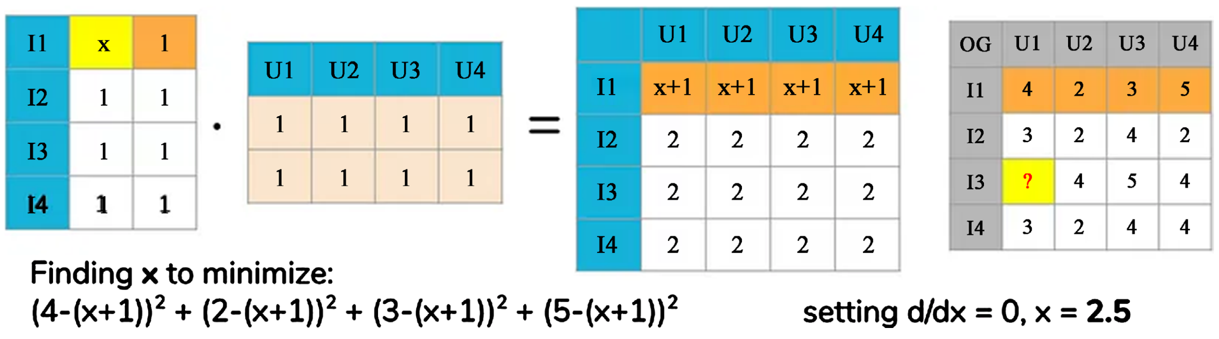
1) The cost function

In this model, the cost function is a measure that allows us to compare the corresponding values in our original utility matrix and our recreated utility matrix. This means that we compare the valuation of U1-I1 in the original utility matrix, 4, with the valuation of U1-I1 in my recreated matrix, 2, and I do the same for all matrix values.

2) Gradient descent with the alternating least squares technique

Gradient descent works by trying to minimize the cost function (RMSE) by changing one value at a time in a component matrix. Let's start by finding the optimal value of the first latent feature of I1 in the component matrix.

Example:



I1	x	1
I2	1	1
I3	1	1
I4	1	1

•

U1	U2	U3	U4
1	1	1	1
1	1	1	1

=

	U1	U2	U3	U4
I1	x+1	x+1	x+1	x+1
I2	2	2	2	2
I3	2	2	2	2
I4	2	2	2	2

OG	U1	U2	U3	U4
I1	4	2	3	5
I2	3	2	4	2
I3	?	4	5	4
I4	3	2	4	4

Finding **x** to minimize:

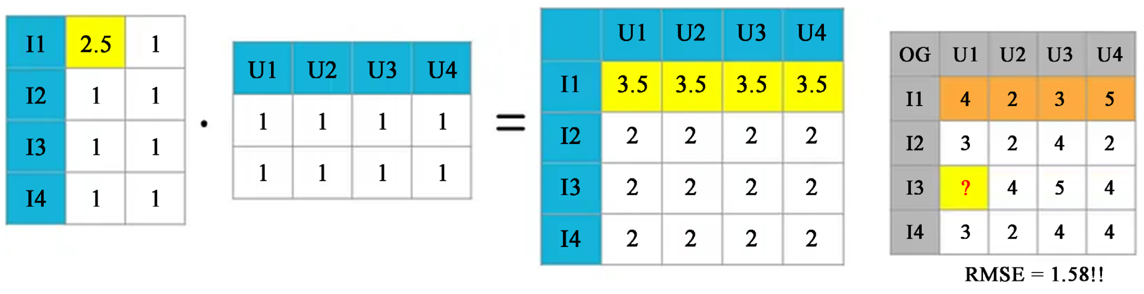
$(4-(x+1))^2 + (2-(x+1))^2 + (3-(x+1))^2 + (5-(x+1))^2$

setting $d/dx = 0$, $x = 2.5$

Figure 8. Minimization of the cost function by the gradient descent method.

As shown in Figure 8, by changing this first value (now denoted **x** unknown), we update the entire first row of our recreated utility matrix. Thanks to matrix multiplication, this whole first row becomes x + 1. Because the rest of the matrix is static, we can simply minimize our cost function on this first row. So we simply minimize this quadratic equation to get an optimal **x** of 2.5.

By replacing the **x** with 2.5, the first row of our recreated utility matrix becomes a row of 3.5, and our RMSE goes from 1.75 to 1.58! (Figure 9) This process is repeated again and again until RMSE cannot improve. It should be noted that changing a value, either in the Item matrix or in the User matrix, changes an entire row or column of the recreated utility matrix. This maintains the relationships between Users and Items, and this process is known as parallelization. By repeating this process over and over again we end up with an RMSE of 1.15, and this is what our recreated utility matrix looks like (Figure 10):



I1	2.5	1
I2	1	1
I3	1	1
I4	1	1

•

U1	U2	U3	U4
1	1	1	1
1	1	1	1

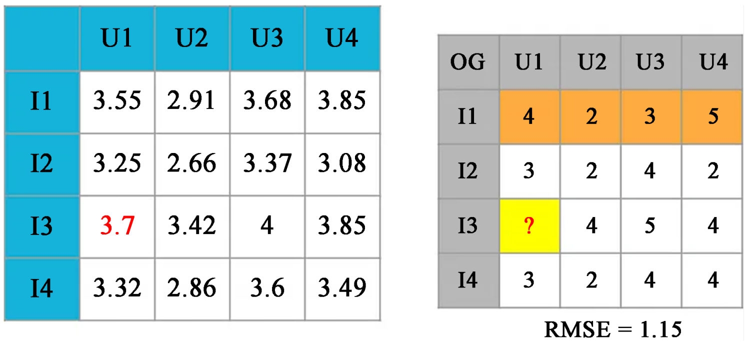
=

	U1	U2	U3	U4
I1	3.5	3.5	3.5	3.5
I2	2	2	2	2
I3	2	2	2	2
I4	2	2	2	2

OG	U1	U2	U3	U4
I1	4	2	3	5
I2	3	2	4	2
I3	?	4	5	4
I4	3	2	4	4

RMSE = 1.58!!

Figure 9. Updating the RMSE.



	U1	U2	U3	U4
I1	3.55	2.91	3.68	3.85
I2	3.25	2.66	3.37	3.08
I3	3.7	3.42	4	3.85
I4	3.32	2.86	3.6	3.49

OG	U1	U2	U3	U4
I1	4	2	3	5
I2	3	2	4	2
I3	?	4	5	4
I4	3	2	4	4

RMSE = 1.15

Figure 10. Updated utility matrix.

3) Evaluation

On this basis, we can guess that the U1 score of I3 is 3.7! In a sparser matrix with several unknown ratings per User, you would then recommend the highest previously unrated Item. Interestingly, compared to our User to User (4.34) and item to Item (3.31) predictions, our SVD value of 3.7 lies between the use of similarities between the two different axes.

In practice, with much more data, one would have to measure the RMSE on the actual evaluation values one has in the test set with their predicted model values. This RMSE would be what is used to evaluate the model, not to be confused with the RMSE used as a cost function in this alternating least squares gradient descent. And this RMSE can be interpreted as the average deviation from the actual score of the predicted score.

3. Hybrid Method: Combining the Two Previous Methods [2] [5] [7]

This approach combines the multiple filtering methods to obtain a refined result.

The principle is to use a user's interests as input to generate a list of recommended products. Many commercial applications rely solely on the products that customers purchase and explicitly rate them to represent their interests, but such systems can also take into account other attributes, including products viewed, demographics and favorite artists. (Figure 11)

In recommender systems, the usefulness of an item is usually represented by a score that indicates how a particular user liked a particular item.

The main problem to be solved is the estimation of scores for items that have not yet been rated by a user. The number of items as well as the number of users in the system can be very large. It is therefore difficult for each user to see all items or for each item to be evaluated by all users. Where it is possible to estimate scores for items not yet rated, items with the highest estimated scores can be recommended to the user.

Although recommender systems can recommend relevant items to a user, they are ineffective when new items are added to the catalogue or when the users are different or new. This cold-start problem is encountered when recommendations are needed for items or users for which we have no information either explicitly or implicitly. There are therefore two cold-start problems: new user and new item.

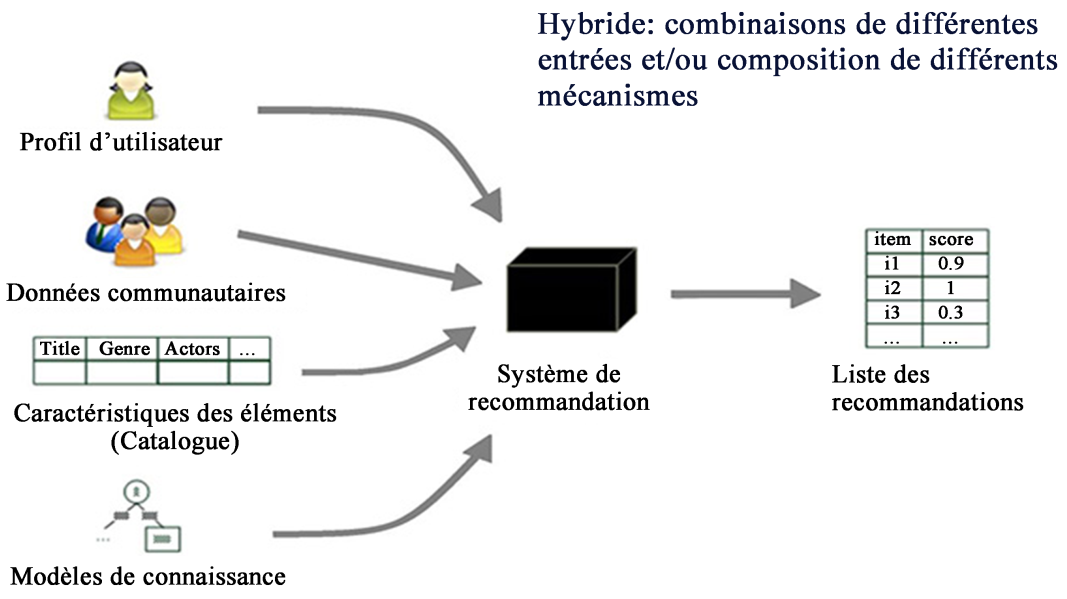

Hybride: combinaisons de différentes entrées et/ou composition de différents mécanismes
Profil d'utilisateur
Données communautaires
Title | Genre | Actors | …
Caractéristiques des éléments (Catalogue)
Modèles de connaissance
Système de recommandation
item | score
i1 | 0.9
i2 | 1
i3 | 0.3
… | …
Liste des recommandations

Figure 11. Hybrid system.

The different criteria for combining the different recommendation filtering techniques are classified as follows:

- Separate implementation of content-based methods and fusion of their predictions;
- Integrating some content-related features in a collaborative approach:
- Integrating some collaborative features into a content-based approach:
- Development of an integrated general model that merges the features of content-based methods and collaborative filtering.

This method reduces filtering problems because the advantages of one technique can be used to minimize the disadvantages of another.

4. Association Rules and Receiver Operating Characteristic Curve (ROC)

An association rule is an implication of the form **A → B** that models the fact that a set of resources **B** is often consumed or accessed when a set of resources **A** has been consumed or accessed. **A** is then called antecedent, and **B** consequent.

An association rule **A → B** has a certain predictive capacity which is measured according to two criteria, called support and confidence. The support **s** of an association rule **A → B** is the number of occurrences of transactions in **D** that contain $A \cup B$.

Normalization has no real use, and is usually used to allow talking in terms of probabilities or percentages. However, when faced with a very large data space, the probabilities and percentages thus obtained become very low, and no longer facilitate reading.

The confidence c of an association rule **A → B** is the conditional probability of transactions containing B knowing that they contain **A** (for a uniform distribution on **D**), *i.e.*, c(A → B) =P(B|A) = s(A → B) s(A).

The first task in using association rules is to discover these rules by searching a database. It is obviously impossible to list all the rules that can be constructed from this data, as the combinatoriality is so great. Consequently, only the rules with the best predictive value are retained. This first step is carried out in two sub-steps:

- Search for rules with a support higher than the predetermined threshold;
- Deduction of the confidence values of these rules and deletion of the rules with a confidence below the predetermined threshold.

To further reduce the number of resulting rules, the notions of closed sets and maximum frequency sets can be used. Closed set A closed set is a set for which there is no superset with the same support. Maximum frequency set A maximum frequency set is a set with a support value greater than the predetermined threshold and for which there is no superset with a support value greater than the predetermined threshold.

Using the notion of a closed set allows the same information to be extracted in a more compact way, whereas using the notion of a maximum frequency set allows the number of rules to be reduced even further but implies a certain loss of information. Several algorithms have been proposed to extract rules according to these two principles. (Figure 12)

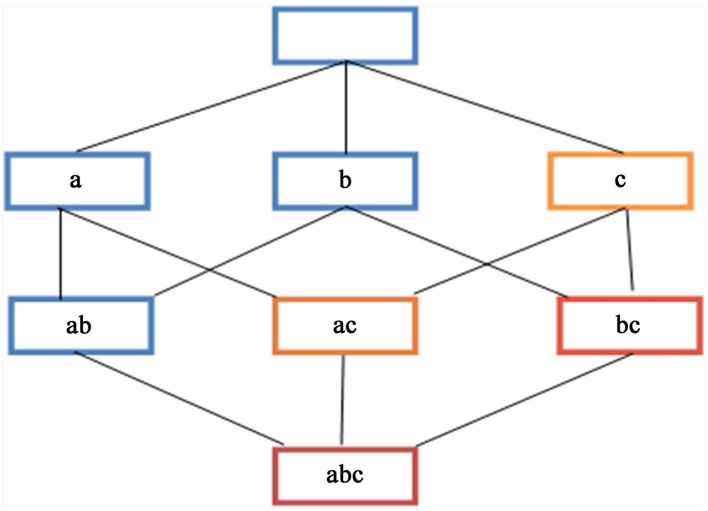


Figure 12. Association rules.

An ROC curve allows the comparison of recommendation algorithms regardless of the quality of the predictions. ROC measures the point at which an information filtering system can successfully distinguish relevant from irrelevant items. In addition, this measure evaluates the order in which recommendations are presented (rank). For this purpose, this curve relates the false positive rates (on the x-axis) to the true positive rates (on the y-axis) in a graph. (Figure 13)

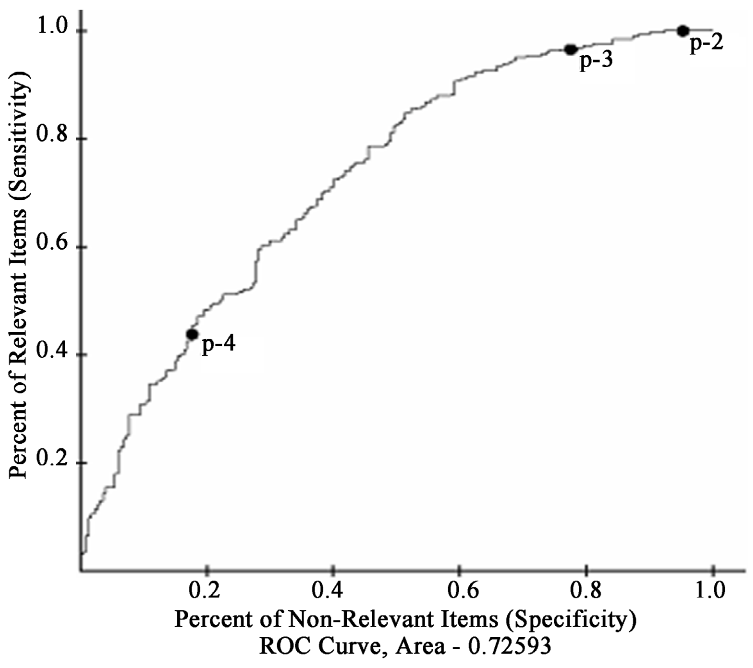


Figure 13. The ROC curve.

From the above, it can be stated that a recommender system performs mainly three actions:

- Extraction of people's preferences from the input data;
- Computing recommendations;
- Presentation to people (users);
- Apart from the cold-start problem, there are two other gaps worth mentioning:
  - Discovery: a list of articles suggested by a Recommender System should allow the user to explore new products as well. Excessive similarity between articles is not very useful in this case and often the user also wants to explore new types of articles compared to the usual ones [8].
  - Filters Bubbles: to identify the information left by the algorithms to the user, E. PARISER introduced the concept of filters bubbles [9]. According to the author, the filter bubble phenomenon is verified when the user finds himself confined in an imaginary bubble built by algorithms and allowing only the passage of certain information. In this way, the user risks creating a partial, not to say limited, view of many facts.

The information analyzed so far does not take into account the context. However, several research studies have shown that contextual information is of paramount importance in the implementation of a recommender system.

The results of various studies have shown a significant difference in rating predictions using relevant and irrelevant contextual information. This allows us to confirm not only the positive but also the significant impact of contextually relevant information on the non-contextualized model.

In order to carry out this work, it seems essential to carry out a study on the online reviews of travelers and/or tourists. The aim is to describe the application of a series of intelligent data analysis techniques to a large number of online travel reviews, in order to automatically extract useful information.

The comments collected from two famous online tourism review platforms, are all those published by customers on specific Italian sites, from 2010 to 2017. A preliminary statistical analysis is performed to gain general knowledge about the subject of the data set, such as the geographical distribution of the reviewers, their activities and comparison between the visit time and the average review score.

Then, natural language processing techniques are applied to extract and compare the most commonly used words on the two platforms. Finally, an association rule learning algorithm is applied to extract favorite destinations from distinct groups of reviewers.

If necessary, the automatically extracted information will be used to create a prototype recommendation system to suggest the best destinations to tourists while taking into account both the constraints related to the cost of the visit ticket and the distance to travel from the user's current location. This will make this work a real market analysis tool for the different service providers.

Since generally "potential travelers tend to rely on the statements of others earlier than on the advertisements of tourism service providers, social networks are an important platform for e-commerce and have one of the most metamorphic impacts on commerce" ([10], p. 2).

Notwithstanding the fact that a lot of information on tourism transactions, customer behaviours, facilities and/or accommodation structures can be found easily and abundantly on the Internet, several studies have been devoted to the analysis of this data. The information obtained can be seen from the point of view of the customer or the service provider. Let's try to get to the bottom of it.

The study of behavioral patterns and user preferences can be useful information for companies insofar as it guides the definition of strategies and offers of added value in marketing to customers but also in the new preferred destinations of the world in the case of tourism.

Since online opinions have the potential to transform the way we do business, working on them, analyzing the motivations of different types of users (buyers of services or sellers) when sharing information and comments online; the impact of the type of sharing on e-commerce are very important ([11], p. 4-5).

From the customer's point of view, the same authors analyzed *TripAdvisor* reviews to implement the useful planning tool for travelers as a decision support system. In the dataset description, the authors address the problems of extracting data on multimodal aspects and consider user-generated photos and text documents to capture correlations between aspects and opinions ([11], p. 5).

More and more users are describing their travel experience on websites. Many comments are generated online every day. This makes it difficult for users to identify useful reviews in a reasonable time frame. Predicting the relevance of reviews allows the user to focus only on the most important ones. This saves time even though usually the implicit assumption is that comments are independent of each other.

In view of what we have just presented, it is now appropriate to implement a prototype recommendation system for tourists. To do this, let us try to analyze the dataset. This dataset contains all the attractions or probable places to visit that are located in the Tuscany region (Italy). Initially saved in .JSON format, the Dataset contains eight attributes, including "ta_id", "name", "ratingValue", "country", "region", "locality", "postal_code", "street_address".

It is not the intention here to analyze each attribute. However, it should be noted that other attributes have been added to the initial dataset. These include: "price", "latitude", "longitude", "altitude", "location".

The geopy function allows us to go from the physical address to the geographical coordinates and/or from the geographical coordinates to the physical address. The Package and Library Pandas, Numpy contain the necessary tools for the implementation of our recommendation system prototype.

The technique adopted in this work is singular value decomposition which applies least squares logic to minimize both the distance and the cost of the ticket. This technique allowed us to highlight the two initial constraints in order to suggest useful destinations to the user for their choice.

The Tourist Visits Recommender System is implemented in the following steps:

1) Decide on the metric (in this case cost and distance minimization);

2) Calculate the distance between the user's actual position and the different locations;

3) Consider the cost of the ticket for each location;

4) Order the locations taking into account both the cost of the ticket and the distance to travel;

5) Select the best results for the chosen metric.

Let's see how this should be done in practice (See Appendix).

5. Conclusions

Our paper has three main axes. In addition to the historical overview, the aim was to expose and analyze the engineering behind any recommender system in order to model, in the last axis of this work, a prototype applicable to tourist visits.

From the noisy and sparse data recorded in .JSON format, we came to create a Dataset in .csv format containing the cleaned data that allowed us to design, model and better implement our *Tourist Visits* prototype to propose attractions (locations or sites) to tourists taking into account the constraints related to cost and distance.

Just as we wanted, the sites located in the client's vicinity and with the lowest cost of access are ordered as shown in the last result. At this stage, we have all the reasons to affirm that our prototype can take into account any other reality and give the expected result. We say that the recommender system suggests useful results.

Conflicts of Interest

The authors declare no conflicts of interest.

References

[1] Benouaret, I. (2017) Un système de recommandation contextuel et composite pour la visite personnalisée des sites culturels. Thèse de doctorat, UTC, Compiègne.

[2] https://interstices.info/les-systemes-de-recommandation-categorisation/#:~:text=Les%20syst%C3%A8mes%20de%20recommandation%20sont,a%20pr%C3%A9f%C3%A9r%C3%A9s%20dans%20le%20passé

[3] https://www.spindox.it/blog/collaborative-filtering

[4] Alchiekh Aldhar, C. (2014) Les systèmes de recommandation à base de la confiance. Thèse de Doctorat, Université de Lausanne.

[5] http://www.cfcopies.com/V2/leg/leg_droi.php

[6] http://www.culture.gouv.fr/culture/infos-pratiques/droits/protection.htm

[7] Resnick, P. and Varian, H.R. (1997) Recommender Systems. *Communications of the ACM*, **40**, 56-58. https://doi.org/10.1145/245108.245121

[8] Shamboura, Q. and Lub, J. (2015) An Effective Recommender System by Unifying User and Item Trust Information for B2B Applications. *Journal of Computer and System Sciences*, **81**, 1110-1126. https://doi.org/10.1016/j.jcss.2014.12.029

[9] Pariser, E. (2011) The Filter Bubble: What the Internet Is Hiding from You? Penguin, UK.

[10] Fazzollari, M. and Petrocchi, M. (2017) Mining Worse and Better Opinions Unsupervised and Agnostic Aggregation of Online Reviews. Institute of Informatics and Telematics, National Research Council, Pisa.

[11] Fazzollari, M. and Petrocchi, M. (2017) Descrizione Dataset Reviewland. Inédit.

Appendix. Application: Prototype Recommendation System

```
# Loading of package, module and library

import pandas as pd
import pandas as np
import geopy
import csv
from geopy import distance

# Function to obtain the geographical coordinates
# [altitude, latitude, longitude] from a physical address

def getlatlong(loc):
    return loc.latitude, loc.longitude

# Loading of Dataset and visualisation of its first five lines

df = pd.read_csv("D:/Christophe/attraction.csv", encoding = "latin-1")
df.head()

# Removal of attributes that do not have considerable entropy here "ta_id".

df_new = df.drop("ta_id", 1)

# Dataset size

print ('We have   ', len(df_new),    'Site in Data)
We have 487 Site in Data

# Data Preprocessing
# Loading the dataset containing the geolocation attributes

data = pd.read_csv("D:/Christophe/attraction_geo.csv")
data.head()

# Adding the "price" attribute

data.insert(5, "price", df["price"], True)
data.head()

# Creation of a dictionary to recursively calculate the distance between the actual position
# and the different attractions [locations].
```

```
with open("D:/Christophe/attraction_geo.csv", "r") as f:
        locations = {    }
        d = csv.reader(f)
        next(d)
        for el in d:

          # Visualiser el

          locations [el[0]] = geopy.location.Location(el[0], (el[1], el[2]))

# Example of how to calculate the distance between two locations

distance.distance(getlatlong(locations['Palazzo Pfanner, Via Degli Asili 33, Lucca, Italia']), getlat-
long(locations['Bagno Chimera, Viale Roma 21 Loc. Fiumetto, Marina di Pietrasanta, Italia']))

Distance(27.321540765341144)

# Creation of a list "myList
# Insert all calculated distances

myList = []
for site in locations:

# print (site)

        dist = distance.distance(getlatlong(locations['Palazzo Pfanner, Via Degli Asili 33, Lucca, Italia']), getlat-
long(locations[site]))
        print (dist)
        myList.append(float(dist.kilometers))

0.0 km
0.38764556260983907 km
0.2509674159974559 km
0.27999638285417033 km
7.222279728441458 km

...

# Removal of attributes that do not add anything

data.drop(['address', 'latitude', 'longitude', 'altitude'], axis = 1, inplace = True)
data.head()
```

```
                                                  location      price

0   Palazzo Pfanner, Via Degli Asili 33, Lucca, It...           15
1   Torre Guinigi, Via Sant'Andrea 45, Lucca, Italia            9
2   Piazza Anfiteatro, None, Lucca, Italia                      0
3   Puccini Museum - Casa natale, Corte San Lorenz...           7
4   Parco Villa Reale, Via Fraga Alta 2, Capannori...           15

# Creation of the "distance" attribute which includes all the distances in the "myList" list

data['distance'] = myList
data.head()

                                                  location      price      distance

0   Palazzo Pfanner, Via Degli Asili 33, Lucca, It...           15         0.000000
1   Torre Guinigi, Via Sant'Andrea 45, Lucca, Italia            9          0.387646
2   Piazza Anfiteatro, None, Lucca, Italia                      0          0.250967
3   Puccini Museum - Casa natale, Corte San Lorenz...           7          0.279996
4   Parco Villa Reale, Via Fraga Alta 2, Capannori...           15         7.222280

# Sorting the dataset in ascending order with respect to the constraints related to
# attributes (["price", "distance"])

data.sort_values(by=['price','distance'], inplace=True)
data.head()

                                                  location   price   distance

52   Domus Romana, Via Cesare Battisti 15, Lucca, I...    0       0.073359
7    Basilica of San Frediano, Piazza San Frediano,...    0       0.144291
282  Chiesa di Santa Maria Corteorlandini, Via S. M...    0       0.144477
2    Piazza Anfiteatro, None, Lucca, Italia               0       0.250967
9    San Michele in Foro, Piazza San Michele, Lucca...    0       0.266617
```