Semiha Gürsoy

A COLLABORATIVE FILTERING SYSTEM FOR PERSONALIZED PRODUCT RECOMMENDATION IN AN ONLINE SUPERMARKET

FINAL MASTER´S PROJECT

Directed by Dr. Aïda Valls Mateu

UNIVERSITAT ROVIRA I VIRGILI

Tarragona
2023

Acknowledgements

First of all, I would like to express my profound gratitude to my Master's thesis tutor, Dr. Aïda Valls Mateu, for her invaluable guidance throughout this project. Her enthusiasm in teaching, support, patience, and understanding have been crucial in this academic endeavor.

I would like to dedicate this work above all to my dear parents, Gülünur & İbrahim Gürsoy. Their unfailing support and belief in me have been the cornerstone of my research journey. In addition to them, I also would like to dedicate this work to all many we tragically lost this year in the earthquake in my beloved country, Turkey.

Abstract

The web platform has become the significant route of trade of the twenty-first century. As the number of options available online, not only the number of sellers but also the products, increased exponentially, the necessity of computer systems to assist users in purchasing decisions has arisen. These systems are known as personalized recommender systems and are mainly developed in the area of Artificial Intelligence. In a broader definition, a recommender system is a software application that actively suggests an item to purchase, to subscribe, to watch, to listen or to invest with the goal of simplifying the decision-making process for the users in the presence of an overwhelming number of options.

This master thesis focuses on the design and implementation of a personalized recommender system for a food supply chain company that operates in Catalonia which specializes in Catalan food industry especially in the meat sector. Utilizing a collaborative filtering approach, this system integrates the users' purchase history and relevant contextual data to generate recommendations. Different levels of a taxonomy of the products have been exploited to identify a set of user profiles, and to select appropriate items to recommend. The experimental results validate the efficacy of the recommender system in identifying potential items of interest for users.

Resum

La plataforma web s'ha convertit en la principal via de comerç del segle XXI. A mesura que la quantitat d'opcions disponibles en línia, no només la quantitat de venedors sinó també els productes, augmenta exponencialment, sorgeix la necessitat de sistemes informàtics per ajudar els usuaris en les decisions de compra. Aquests sistemes es coneixen com a sistemes de recomanació personalitzats i es desenvolupen principalment en l'àrea de la Intel·ligència Artificial. En una definició més àmplia, un sistema de recomanació és un programari que suggereix activament un article per comprar, subscriure's, mirar, escoltar o invertir, amb l'objectiu de simplificar el procés de presa de decisions per als usuaris en presència d'un nombre aclaparador d'opcions.

Aquest treball de fi de màster se centra en el disseny i implementació d' un sistema de recomanació personalitzat per a una empresa de subministrament d' aliments que opera a Catalunya i que està especialitzada en la indústria d' alimentació catalana, especialment en el sector càrnic. Utilitzant un enfocament de filtratge col·laboratiu, aquest sistema integra l'historial de compres dels usuaris i les dades contextuals rellevants per generar recomanacions. S' ha fet servir la informació de diferents nivells d' una taxonomia dels productes per identificar un conjunt de perfils d' usuari i seleccionar els elements apropiats per recomanar. Els resultats experimentals validen l' eficàcia del sistema de recomanació per identificar elements potencialment interessants per als usuaris.

Resumen

La plataforma web se ha convertido en la principal vía de comercio del siglo XXI. A medida que la cantidad de opciones disponibles en línea, no solo la cantidad de vendedores sino también los productos, aumenta exponencialmente, surge la necesidad de sistemas informáticos para ayudar a los usuarios en las decisiones de compra. Estos sistemas se conocen como sistemas de recomendación personalizados y se desarrollan principalmente en el área de la Inteligencia Artificial. En una definición más amplia, un sistema de recomendación es una aplicación de software que sugiere activamente un artículo para comprar, suscribirse, mirar, escuchar o invertir, con el objetivo de simplificar el proceso de toma de decisiones para los usuarios en presencia de un número abrumador de opciones.

Este trabajo de fin de máster se centra en el diseño e implementación de un sistema de recomendación personalizado para una empresa de suministro de alimentos que opera en Cataluña y que está especializada en la industria de alimentación catalana, especialmente en el sector cárnico. Utilizando un enfoque de filtrado colaborativo, este sistema integra el historial de compras de los usuarios y los datos contextuales relevantes para generar recomendaciones. Se ha usado la información de diferentes niveles de una taxonomía de los productos para identificar un conjunto de perfiles de usuario y seleccionar los elementos apropiados para recomendar. Los resultados experimentales validan la eficacia del sistema de recomendación para identificar elementos potenciales de interés para los usuarios.

Index

1.Introduction

1.1.Motivation

The e-commerce platforms provide consumers with a huge amount of options to evaluate and decide from. In order to overcome the information overload and help the users in making their decision, personalized recommender systems are built to offer better user experience. This kind of system is also a benefit for the company, increasing the sales. Personalization refers to providing individual services to each one of the customers for identifying individual preferences and meeting their needs as one of the effective strategies in marketing [1].

Considering the limited screen sizes and the limited attention span given by the users while navigating online [2], the businesses are required to make an accurate and personalized filtering of products in order to engage the customers [1][3].

The Artificial Intelligence (AI) field has a long tradition in the study of the development of personalized recommender systems. However, this problem still has open questions and the applicability of the existing techniques to real environments generates a lot of challenges, as each e-commerce system has different characteristics.

In this master thesis, we will design and implement a personalized recommender system for a food supply company that works in Catalonia. This company has provided some support to the development of this master thesis, by facilitating knowledge and information about their shopping models. We propose to follow a collaborative recommendation approach [2][6], based on both the lists of products bought by the clients, some demographic information and some contextual data.

Simply put, recommender systems that are using collaborative filtering first identify groups of users in taste space (they buy similar types of products in similar conditions), thus, building some generic user profiles. Then, to make a recommendation to a user, it is first assigned to one of the profiles and the information of the products bought by the members of the cluster are used to generate a personalized list of recommendations [6].

1.2.Objectives

The objective of this work is to design and implement a personalized recommender system for a food supply chain company that operates in Catalonia. This general objective can be decomposed in the following specific objectives:

- Research on previous work done in collaborative filtering recommender systems in the ecommerce context
- Preprocessing of the available data of historic sequence of purchases done by clients to construct a user-item matrix
- Creating the customer profiles using AI-based clustering algorithms
- Calculating the similarity of users
- Exploitation of the clusters to generate a short list of recommended products for a certain customer.

1.3. Document Structure

The rest of this document has the following structure. Chapter 2 makes a short introduction to recommender systems in Artificial Intelligence, and reviews the related works about e-commerce recommendation. Chapter 3 proposes the design of a collaborative recommender system for a particular food supply company. Chapter 4 explains the implementation of the system step by step. Chapter 5 offers the obtained results and Chapter 6 gives our conclusions.

2.Background and Related Work

In this chapter, we provide a brief overview of the relevant researches and fields of application. We discuss the most important problem domains, introduce the related terminology and point out the basic approaches.

2.1.Background

The personalized recommender systems are built on the idea of profiling the users and the products and relating them. The first step is to create a user-item interaction matrix to represent the interests of the users on corresponding products individually. Each user is represented by a row in this matrix and each item (such as a movie, product, song) is represented by a column. If a user has interacted with an item then the corresponding cell in the matrix is filled in the way this interaction is preferred to be represented. If a user has not interacted with an item, the corresponding cell is left blank or filled with a neutral value. Thus, each row in this matrix can be thought of as a user profile.

There are two approaches to fill this matrix by using explicit input and by using implicit input. The explicit input refers to the ratings directly given by the user to the respective product. For example, Netflix collects star ratings for movies and TiVo users indicate their preferences for TV shows by hitting thumbs-up/down buttons[3]. The implicit input refers to the ratings or representation of interest or opinion of a user on a product indirectly. Types of implicit feedback include purchase history, browsing history, search patterns, or even mouse movements.[3] By introducing these inputs obtained directly or indirectly, into the user item matrix we obtain a representation of the user profiles.

To generate recommendations based on these user profiles there are two main approaches: content-based filtering and collaborative filtering. These are called filtering as they are the ways of filtering huge amounts of data for the corresponding user profile. The former makes recommendations by utilizing the characteristics of items the users have interacted with in the past. If a user often rates horror movies highly the system might recommend more horror movies to them. Content based filtering does not create “collective user profiles”. Instead each item is treated as it is an individual profile based on its characteristics and the items are recommended based on how closely their profiles match the user’s preferences.

Collaborative filtering has two approaches: item based collaborative filtering and user based collaborative filtering. On the contrary to content based filtering, similar items or users are identified based on their interactions. For example, in item based collaborative filtering if two items were rated similarly by many users they are considered as similar items. In the user based collaborative filtering, the users A and B are considered to be similar if they have interacted with or rated many of the items

similarly so the recommendations are done based on the similarity between the users. In this sense, the user based collaborative filtering makes the recommendations based on the “collective opinions” of users and allows us to consider the opinions of thousands[2].

2.2.Related Work

Collaborative filtering is a rapidly advancing research area[4]. We can divide the collaborative techniques as classic techniques and more recent techniques.

The classic techniques include neighborhood methods such as memory based or user based collaborative filtering[4] which focus on finding the neighboring user profiles. The more recent techniques are often utilizing the matrix factorization methods such as singular value decomposition and non negative matrix factorization[4] for addressing data sparsity problem.

Other approaches such as clustering with the intention of grouping the users of similar preferences to increase the efficiency of recommendations or deep learning to capture complex user-item interactions to overcome scalability issues in large datasets and hybrid systems which combine collaborative and content based filtering are offered as well.

Taking into account that the effectiveness of these techniques varies depending on the specific context and implementation in this work we have focused on user based collaborative filtering and took [1] as a road map of our implementation.

2.3. Practical Challenges of User-Based Collaborative Filtering

Despite it simulates how humans share their opinions with others[2] and is often found to be successful[1] there are practical challenges the user based collaborative filtering faces:

- Sparsity of the user-item matrix: There are a vast number of users and not every user in the system has interacted with each item which causes the user item matrix to have very big dimensions with sparse information.
- Cold start: The system can not recommend an item to a user who has recently joined the recommender system since there is not enough information about him or her.
- Scalability: Due to sheer volumes of data the system has to spend a lot of time and resources such as processing power or memory in order to extract the similarities among all the customers which causes poor productivity.
- Change in user preferences: In the context of this work which is based on a supermarket’s product domain we have not encountered a major difference in the purchase patterns. However, the change in user preferences often is a challenge for the systems which use user based collaborative filtering in the industries with rapidly changing product domains such as streaming

platforms, or fashion based industries.

In our implementation we used the hierarchical category information of the products rather than the products themselves in the user-item interaction matrix with the aim of overcoming the sparsity challenge and preferred to apply clustering and caching in order to overcome the scalability challenge. We did not implement a specific functionality for the cold start problem and also did not take the change in user preferences into account. The detailed explanations of the implementations are given in the following section.

3.Design

In this chapter, we provide a detailed explanation of our design choices about all the steps of the project.

3.1. Design Choices Regarding the Data to be Used

As we build this system by utilizing the real data obtained from a company, we made our design decisions taking the characteristics and limitations of the data. We obtained a total of 406 MB data in four files, which corresponds to the purchase history of 62590 users' between the dates 01-01-2020 and 03-04-2023.

The data obtained does not contain the rankings on items nor item characteristics which eliminated the content-based filtering option for us. Taking into account that there are 22009 items and 62590 users, we put our focus on user based collaborative filtering.

Products are organized by the company in different categories. Moreover, there is a category hierarchical tree that groups the specific categories into more general ones. For example, we can see a portion of the hierarchy of categories of products in Figure 1. The figure visualizes some of the categories and their respective levels in our implementation as well as the parent-child relationship between them.

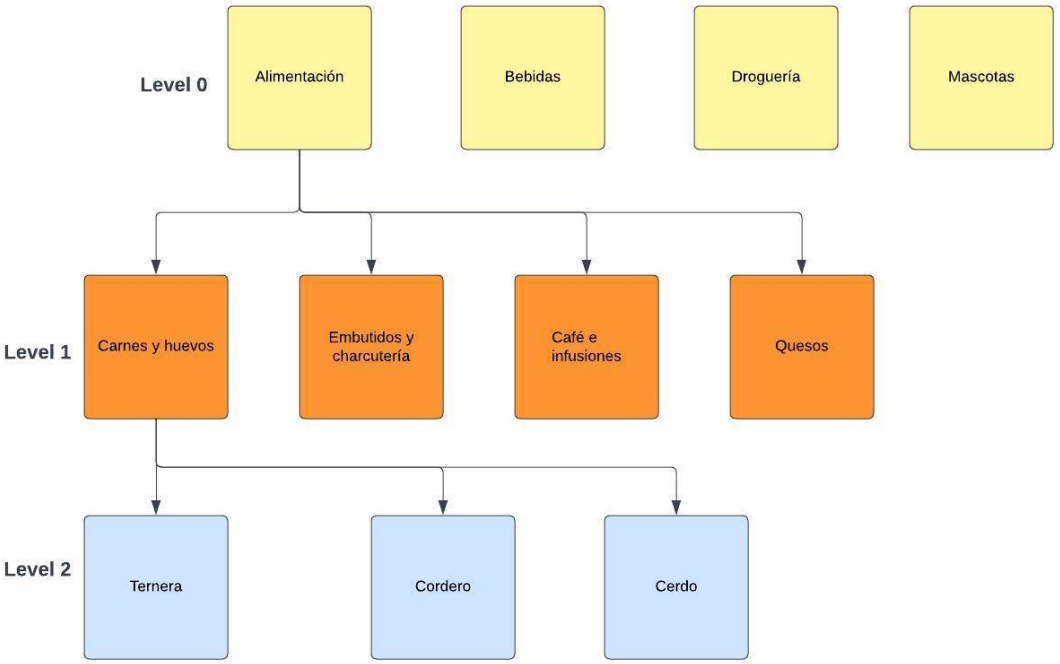

Figure 1. An example on category hierarchy

The significant processing power and memory that would be required to operate on a user item matrix of such large dimensions (62590 x 22009) and the hierarchical category information we had along with the articles (products) prompted us to the decision to create a user-category interaction matrix in order to reduce the dimensions, instead of creating user item matrix. Matrices for different category levels have been built and studied, as will be explained later.

Another decision we took to further reduce the dimensions of the user category interaction matrix was to apply clustering on the users in early stages of the recommendation process. We then preferred to apply the collaborative filtering on these user clusters with the aim of representing the users' taste space in smaller groups and made our recommendations to the users based on the taste group they belong to.

Our design choices were mainly made based on the amount and the limitations of the data. The following section gives details about the data, its preprocessing phase and the challenges we have faced.

Regarding the programming language, the project has decided to be done using Java programming language version 17, as it was appropriate for the company, and it also has libraries that implement Machine Learning methods and Recommendation tools that are useful for this work. Considering the vast amount of data and the limitations of manipulating it in programs such as excel the MySQL database management system is decided to be employed to store, retrieve, analyze and manipulate the data in a fast manner.

3.2.Design of Data Preparation

The data was obtained in .csv format in four separate files:

- articulos.csv : 22009 lines in total containing the article code, description, hierarchical category information such as
  article code; Redondo atado de ternera;Alimentación > Carnes y huevos > Ternera
- clientes.csv : 62590 lines in total containing the client id, postal code and year of birth such as
  client id;25300;1987
- pedidos.csv : 564612 lines in total containing the order information with order id, client id, order creation date, order creation hour, order service date such as
  order id;client id;5/1/2020;9:33:13 AM;7/1/2020
- lineaspedidos.csv : 16656305 lines in total containing the order detail information with order id, article id, quantity, price per unit such as
  order id;article id;1;1,23

In order to store, access and filter the data easily we decided to create a MySQL database with a better organization of the tables in order to optimize the creation of the user-category matrix. The database would consist of five tables. The designed database schema is given in the ER diagram below:

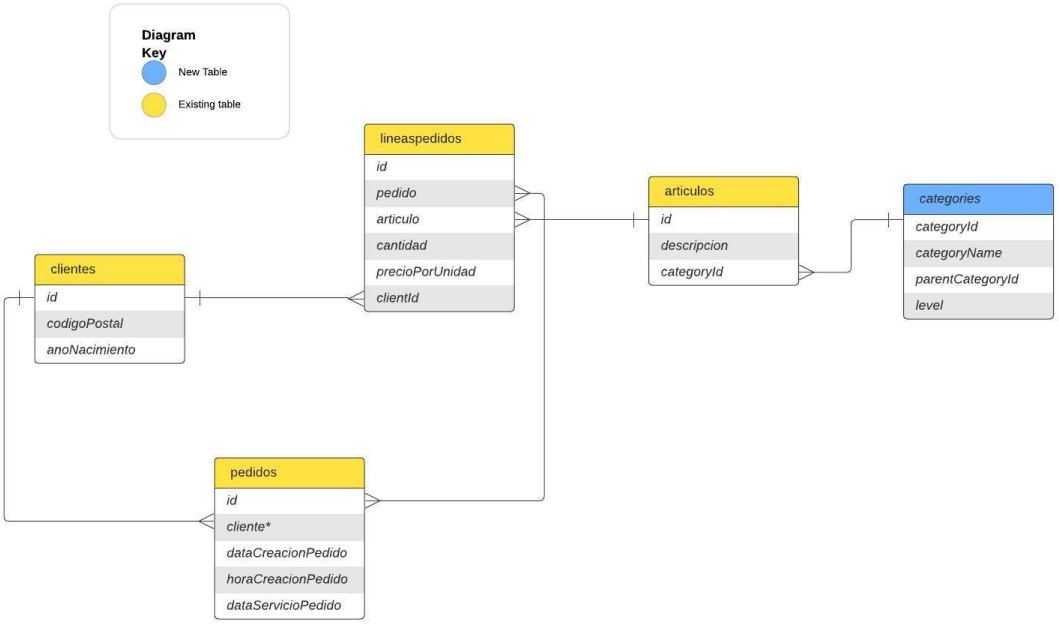

Figure 2, ER diagram of proposed database

In this ER diagram the yellow color represents the existing tables, referring to the csv files we obtained, the blue color represents the new table we aim to create. The sides of the entity relationship lines with the fork represents the “many” side of the entries meaning that there can be many entries with the given data, the other side of the lines with a little vertical line represents the “1” side of the relationship of

entities meaning that there can be only 1 entry of given data. Such as there can be only 1 client with the given “id” however there can be many order detail entries which have the same client id.

3.2.1.Design of Data format

Following the proposed database design, we plan to have the data stored in the respective formats given in the table below:

Table Name	Column Name	Data Type
articulos	id	integer
	codiArticle	varchar
	descripcio	varchar
	categoryId	integer
clientes	id	integer
	codigoPostal	varchar
	anoNacimiento	smallinteger
pedidos	id	integer
	cliente	integer
	dataCreacionPedido	date
	horaCreacionPedido	time
	dataServicioPedido	date
lineaspedidos	id	integer
	pedido	integer
	articulo	integer
	cantidad	integer
	precioPorUnidad	float
	clientId**	integer
categories	categoryId	integer
	categoryName	varchar
	parentCategoryId	integer
	level	integer

Table 1. The tables and their corresponding columns along with the data type stored

3.2.2.Design of Category Identification and Hierarchy Construction

The hierarchical category information were given in each line of the article file, which are of the form:

```
artileID; Redondo atado de ternera; Alimentación > Carnes y huevos > Ternera
```

In our design, the program extracts the category information of each product from the string line and builds a category table entry that stores the category information

by their level as well as the links between these levels. In this example, “Alimentación > Carnes y huevos > Ternera” represents the breadcrumb of the particular article and the innermost level is “Ternera”, its parent category is “Carnes y huevos” and its parent category is “Alimentación”. Therefore, “Alimentación” should have the level 0, the outermost level category and its parent is NULL as it is the root; the categories “Carnes y huevos” and “Ternera” are level 1 and level 2 respectively. This way we aimed to discover the hierarchy of categories while reading and storing the product's information.

By linking the current category and their parents with their ids in a recursive way, we also aim to benefit from fast access to the upper levels of categories as well as filtering the categories by specifying their levels or their parents.

3.2.3.Database Query Path

A limitation to take into account when having a denormalized database structure is the possibility of defining incorrect paths for data retrieval. For example, if we start by retrieving the clients and for each client retrieving the order ids, with the client id search the articles in the order details, and finally with the order id and the category ids in the articles get the category id. This path is circular and not optimal, especially considering that there are 16656306 product interactions (i.e. article purchases). Figure 5 shows this circular inefficient path.

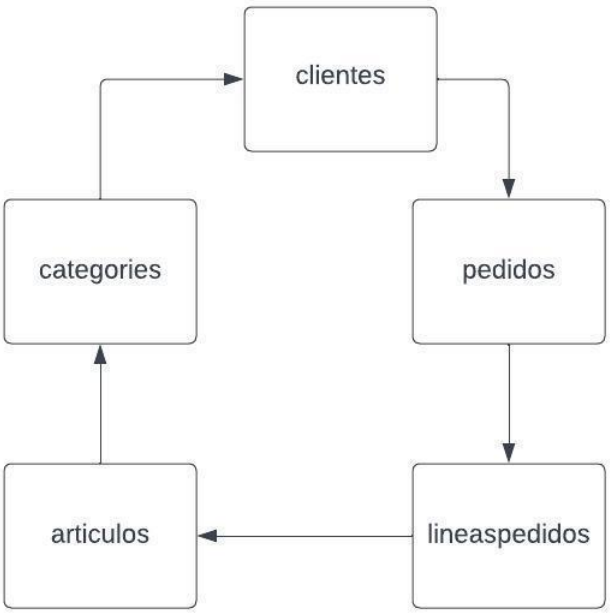

Figure 3. The path followed between the database tables in order to retrieve user - category interactions

The proposed path takes as initial data retrieval point the lineaspedidos table. Then, we would have the benefit of reading a single line that had the order id, article id and the quantity purchased for that specific order however we were missing the client id. By utilizing the order table, we planned to join the pedidos and lineaspedidos table in

order to create a new clientId column to the lineaspedidos table. Which in the end would enable us to read in only 1 line the client, the article and the quantity of purchase for the respective article.

Taking into account that for the procedure proposed, the lineaspedidos table would store the article ids and the client ids with the foreign key linking between the articulos and clientes table and we would benefit from this indexing. With the foreign key links we would be able to eliminate many order detail entries that did not have a corresponding user or corresponding order entries.

3.2.4.Decisions on Handling the Missing Values

The foreign key links which were explained in the previous section, can only be created if for each line of table that will hold the foreign key there is a corresponding entry on the table to be linked. So while linking the lineaspedidos (order details) and pedidos (orders) table, the entries which did not have a corresponding order id in the pedidos table would need to be deleted.

As our main idea is to work on the user-category interaction matrix, the article entries that did not have the category information would need to be deleted as well as they will not be linked to a category even though a user has purchased them. The figure shows us some articles that need to be deleted.

Hammerite Superia 750 ml Forja Negro 5379659	Bricolaje > Pintura > Tratamiento hierro
Radiador Aceite Orbegozo RN2000 2000W Negro	Electrodomésticos > Calefacción, ventilación y extracción > Calefactores y radiadores
QUESO SEMICURADO CORAZON DEL LUGAR PIEZA	
QUESO TIERNO BARRA MANZER 3 KG	
CUARTO CORDERO DELANTERO	
CREMA DE QUESO VILLA CORONA CUBO 2 KG	
CREMA DE YOGUR VILLA CORONA CUBO 3,5 KG	
QUESO FRESCO BURGOS VILLA CORONA NATURAL 1 KG	
QUESO EDAM LONCHAS QUELAC 1 KG	
QUESO RALLADO EMMENTAL QUELAC 1 KG	
MOZZARELLA RALLADA QUELAC BOLSA 1KG	
Pure de Patatas Promolac 1 KG	
MANTEQUILLA ITALIANA ZANETTI 1 KG Italiana Zanetti	
Lengueta P/Boca D.55 'T' TRK02001 BMM	Material ganadero > Cerdo > Transportadores pienso
Snacks de secallona	Alimentación > Embutidos y charcutería > Curados > Pieza
Clim fibra extreme antibacterial	Droguería > Utensilios de limpieza > Estropajos
Bruguerdux Brill. 750ml Azul Ancla 5159289	Bricolaje > Pintura > Esmaltes > Brillante
Contenedor alimentario hermético 0.5l 182288	Hogar y jardín > Hogar y menaje > Contenedores alimentarios
Bateria Psion WAP4 WA3010 4400mAh	Informática y papelería > Material tiendas y negocios > TPV > Artículos TPV
Medidor Plast.Graduada 2L Pico Flex.MPF-2 Faherma	Automoción > Bidones almacenamiento

Figure 4, A portion from the articles.csv file demonstrating the articles need to be deleted

We also detected that some of the rows in the users table did not have year of birth or the postal code. We intended to build the user clusters based on their demographics to study the change in recommendations based on user demography and due to this reason, the rows that were missing the data would need to be deleted. The figure shows us some example data from the client entries that need to be deleted.

2	3
CODIGO POSTAL	AÑO NACIMIENTO
8960	
25210	
25270	1966
8950	1982
8711	1973
50170	1954
8005	1978
25300	1978
25520	1964
8759	
8490	1965
8670	1964
25280	1976
25230	
25001	
8830	1973
25300	1978
25310	

Figure 5, A portion from the clientes.csv file demonstrating the clients need to be deleted

3.2.5.Decisions on Handling the Anomalous Values

It is also necessary to point out that while working with real data, the data preprocessing design should include the strategies of handling the anomalous data as well. These entries can seem to be fine as they would not cause parsing or format exceptions during the implementation. However, the existence of anomalous data would yield misleading values once the implementation is realized. Our strategy to handle such data is to eliminate the rows that include the incorrect data and their linked entries from the entire dataset. Figure shows us the anomaly on the year of birth column in the client entries.

anoNacimiento
0
0
0

Figure 6, A portion from the clientes.csv file demonstrating the anomalous data

3.2.6.Design of User-Category Matrix Creation

During the construction of the user-category matrix, a significant limitation we possibly would face about extracting the data would be the high read time required

as a natural result of working with a very big database with thousands of products and users. In the settings where there are continuous changes on data, this limitation could require more processing power or parallelization of the extraction tasks. In our case, since we will build the recommender system with a fixed sample of the data, we preferred to extract and create the user item matrices for different category levels and store them in permanent Java object serializable files, to benefit from fast reading whenever necessary.

Observing the dataset, we realized that the number of products bought from a certain category were between 0 and 15. However, we encountered some instances with more than 50 purchases in one of the categories. This fact showed us the necessity of applying normalization before using clustering techniques. From several techniques available, following reasons lead us to use min-max normalization:

- We did not know the actual range of the purchases in a single category.
- We needed to avoid the higher purchase instances dominating the smaller instances.

After deciding the normalization step, we moved forward to design the recommender system.

3.3.Design of the Recommender System

This section gives detailed explanation about the design of steps to be taken in order to implement the collaborative recommender system. The possible methods, algorithms and approaches could be employed are explained and the discussion over making the most adequate selection of these techniques is made.

3.3.1. Technologies Used

The project is planned to be built using Java programming language version 17. Regarding the libraries that implement Machine Learning methods, we preferred to use WEKA for clustering for the ease of use and the extensive documentation it provides. However WEKA does not directly support collaborative filtering as it is more of a general purpose machine learning tool. Out of many libraries we have searched, taking into account that we require compatibility with java, and the availability of the similarity measures adequate for our case which are cosine similarity and pearson correlation are commonly supported by both WEKA and Apache Mahout, we decided to use Apache Mahout to apply collaborative filtering. Other advantages of using Apache Mahout in such a project are the extensive documentation it provides, the availability of many examples and it does not apply in our case but the compatibility of working in distributed computing environments such as cloud systems.

The methods that were implemented by us as well as the reasons for choosing them along with their outputs are given in the following sections.

3.3.2. Design of the Recommendation Procedure

The method of recommendation proposed in this master thesis consists on the following steps:

1. Clustering: Creation of a few general user profiles. This will be done by applying an unsupervised clustering algorithm to the user-category interaction matrix.
2. Collaborative Filtering: Comparison of users in the clusters generated, by calculating the pairwise similarity of them.
3. Neighboring: Using the obtained results of similarities among the users in the same cluster and given a user ID, detecting the most similar users to him/her.
4. Recommendation: Obtaining the product list of recommendations for the client, given his user ID.

The detailed design choices regarding the recommender system are given in the following sections.

3.3.2.1. Clustering

We planned to apply clustering on the user-category matrix as a strategy to reduce the vast dimensionality in the matrix with regards to the number of users. Having a reduced number of clusters of similar users permits to speed up the comparison of a new user with profiles of existing ones.

Taking the speed as a significant computing limitation and considering that some of the clustering algorithms are specifically working on item characteristics which is a type of data we do not possessed, from the possible options of K-Means, Hierarchical Clustering, Expectation-Maximization or Cobweb we preferred to employ the KMeans algorithm and used the respective function of WEKA Library to obtain the clusters.

When working with the K-Means algorithm, a crucial point is to find the optimal number of clusters and in order to find it we need to run the clustering on 3 user-category matrices we created and evaluate the score. From the available evaluation metrics of Within Sum of Squares (WSS), Between Sum of Squares (BSS), Silhouette Score we eliminated the Silhouette Score due to the computational complexity it has which comes from computing the square of distances between all the elements in the clusters and the processing time it would require. We preferred to calculate a proportion between the SSbetween/SSwithin considering they are the measures of:

- SSbetween : measure of the variability between the clusters. It is essentially a measure of *how different each cluster is from every other cluster*. When

SSbetween is high, it means the clusters are well-separated from each other, implying that the clustering solution is doing a good job of partitioning the data into distinct groups. In general, the higher the SS-between, the better the clustering solution.

- SS-within: measure of the variability within each cluster. This is essentially a measure of *how similar the observations within each cluster are to each other*. When SSwithin is low, it means that the observations within each cluster are very similar to each other, implying that the clustering solution is doing a good job of grouping similar observations together. In general, the lower the SSwithin, the better the clustering solution.

In other words, a good clustering solution needs to have a high SSbetween, showing that the clusters are distinct from each other and a low SSwithin, showing that the observations within each cluster are similar to each other. These two measures help to capture the two main goals of clustering: to make sure that things within each cluster are as similar as possible (low SSwithin) and that each cluster is as different as possible from the others (high SSbetween). Thus the necessity for calculating a ratio of them in order to determine which number of clusters would be the most adequate arised.

3.3.2.2.Collaborative Filtering

In the clustering phase, we planned to obtain the groups of users that have similarity within their groups. During the collaborative filtering stage, our aim is to identify the most similar users within the cluster they belong to. There are several measures to calculate similarity, being the following ones the most used:

- Cosine Similarity: Measures the cosine of the angle between two vectors. It's often used in text mining and works well when the magnitude of the vectors is less important than the direction of the vectors. Adapting this explanation to our case; if the total count of all items a user has interacted with were less important than the relative preferences of the user, or the patterns of purchase of the user we would use the cosine similarity, which in our case the total count of interactions representing the interaction with the categories is crucial as we aim to recommend products within the same category. Therefore, cosine similarity was not the ideal method for our purpose.

- Pearson Correlation: Measures the linear correlation between two sets of data. It normalizes the effect of the mean and scales so that the values lie between -1 & 1. In the context of a recommender system, it can be used to measure how similar two users are in terms of their rating/purchase patterns. In our case, the number of interactions with each category can be thought of as how much they "like" or "prefer" that specific category. The Pearson Correlation in this context gives a measure of how similar two users are in terms of their category preferences.

In our designed system for instance, out of three customers A, B and C, whose number of interactions with the same category are 10, 8 and 2 respectively, then users A and B should have more similarity than C. Cosine similarity considers only the orientation; however the Pearson correlation takes both orientation and the magnitudes into account. Therefore, we decided to use the Pearson correlation in order to calculate the similarities between the users.

3.3.2.3.Neighbouring

Till this point we have decided that for each cluster obtained, the pairwise similarities among users using the Pearson correlation would be calculated. Our next aim in this case, would be to identify the top 'n' users with the highest similarity for a given user ID. To achieve this, out of many algorithms possible, due to its effectiveness and simplicity we decided to employ the K Nearest Neighbor (KNN) algorithm.

While in our case no new data were introduced to the system, and thus, user similarities remained static, the KNN algorithm would indeed be an adequate choice in a dynamic context where user data evolves over time. The K Nearest Neighbor algorithm's ability to adjust and update the list of top similar users without needing to recalibrate the entire model is a significant advantage in dynamic environments.

3.3.2.4.Recommendation

In order to create a list of products recommendations we must consider several factors, which are:

- In the context of the products we buy from a supermarket, our purchase patterns generally remain consistent over time. We tend to purchase similar products, in similar quantities. Thus, it is logical and necessary (from the evaluation point, explained in the evaluation section) to include some products purchased by the client into the recommendation list. Out of the "N" number of products to be recommended, at least a portion should represent the "taste" of the client in question. In order to include variety to this specific portion, randomization product retrieval can be applied.
- In the "Neighboring" section, we aimed to retrieve the IDs of the most similar users to the user in question. Using these IDs, our plan is to retrieve products from the database that were purchased by the neighbor users. Again, for the diversity of the product, randomization can be used.
- Due to the large number of different products available in the shop, we can observe that users only interact with a subset of the categories. When, using level 1 or level 2 in the category tree, we can also observe that for some categories the number of purchases made is quite low. These two facts open the possibility of a low number of products to be recommended from the purchases of the neighboring users. That is, if the neighboring user made a purchase from category x but he purchased only 2 products, we would be recommending only these two products. This low number conducted us to

evaluate the sufficiency of the “number” of elements in the recommendation list. To solve this problem we propose the following:

- Having a “fixed number” for the recommendation list. Thus, setting the length of the list apriori.
- Calculating the sum of the number of products recommended by utilizing the taste space of the user and the neighboring user preferences. If the sum is lower than the list size, we need to fill it.
- In such a scenario when the list could not be filled, from the products provided by utilizing the taste space of neighboring users find their category and retrieve N random products in order to fill the list.By including randomization within the borders of the category interacted with, we aim to have diversity in the products. That is, if the category in question is “Cheese”, we extract some different products of “Cheese”, even if it is not in the neighbor user list.

Combining these three factors, we design the system to recommend a fixed number of products in each run, which initially be divided into two portions: the products which represent the preference of the user in question and the products which represent the collective idea of the neighboring users and in case these are not enough populating the list with the random items within the borders of interacted categories.

It would be necessary to remind here that utilization of user-category matrices of different levels of categories would yield recommendations of different product spectrum. We would like to explain this scenario with an example. The following table shows an example category hierarchy, their respective levels and some of the sub-categories that are their descendants:

Category Name	Level	Some sub-categories
Alimentación (Food)	0	Pan(Bread) Salsas(Sauces)
Lácteos y derivados (Milk and derivatives)	1	Leche (Milk) Yogures(Yogurts)
Leche (Milk)	2	-

Table 2. An example of the hierarchical category structure

If the interaction is made with the “Leche” (Milk) which is in level 2, within the user-category matrix of level 2 this specific category would be utilized. However for the user-category matrix of level 1 this interaction would be considered to be made under the category of Lácteos y derivados(Milk and derivatives) which would conclude other interactions made with Yogures(Yogurts) category as well. When the level 1 of the user-category matrix is used, there is a possibility of a yogurt product to be recommended to you even though you purchased milk before. This variation in

the product spectrum is a natural result of collecting interactions under a higher level of category. In this context, when the milk is purchased but we are utilizing the level 0 of the user-category matrix there is also a possibility for watermelon to be recommended as well. In some contexts, the recommendation of watermelon is more logical than the recommendation of kitchen towel when the past purchase is milk yet the level of category which would give the most accurate results in general can be discussed over the test results. The tests run in order to evaluate the accuracy of the system are given in chapter 5.

3.3.2.5.Design of Evaluation

There are several evaluation methods that are specifically designed for evaluating the recommender systems. From our research, some of these methods and their brief descriptions are given in the table below:

Metric	Description
Precision@k	The proportion of recommended items in the top-k set that are relevant.
Recall@k	The proportion of relevant items that are found in the top-k recommendations.
F1 Score	The harmonic mean of precision and recall.
NDCG (Normalized Discounted Cumulative Gain)	It considers the ranking of the recommended items. It assigns more importance to relevant items appearing at the top of the list.
RMSE (Root Mean Square Error)	Could be used in rating prediction tasks. It measures the differences between the predicted and actual ratings. *we do not have ratings
MAE (Mean Absolute Error)	Error metric that calculates the absolute difference between the predicted and actual ratings. *we do not have ratings

Table 3.Possible evaluation metrics and their descriptions.

Taking into account that in our dataset there are no ratings per product, we moved our focus to the evaluation metrics which calculate the proportion of relevant products over all recommendations. The “relevance” in this context refers to the products purchased by the user in the past as they are the direct representation of

their preferences. In this context out of the first two options Precision@k and Recall@k:

- Precision@k: The proportion of recommended items in the top-k set that are considered to be relevant. A high precision@k signifies that most of the products recommended to the users are indeed relevant, however it does not account for all the relevant products that might have been missed.
- Recall@k: The proportion of relevant items that are found in the top-k recommendations are calculated. A high recall@k means that most of the relevant items were recommended to the users, but it doesn't account for irrelevant items that might have been recommended as well.

Considering that Precision@k prioritizes the quality of the recommendations whereas Recall@k prioritizes the quantity and we have designed to offer a fixed number of recommendations per run we decided to choose with the Precision@k as evaluation metric.

The “k” of the Precision@k metric refers to the number of top recommendations that are relevant. The optimal number of k could be found by running several tests. The experiments conducted to determine the optimal 'k', their evaluation, and subsequent discussions are presented in chapter 4.

As every metric has its limitations and potential biases, one potential bias the Precision@k might introduce to our system is that in a fixed number of recommendations and fixed number of relevant items (which is the k of the Precision@k metric) there is a possibility of the evaluation to have the same proportion. This means that if the top 'k' products are selected from 'n' recommendations every time the system runs, and there are consistently 'k' relevant products recommended, this ratio would remain constant at 'k/n', even if the actual recommended products change. In this case we could observe and evaluate the relevance of the recommendations by the nature of the product however the precision@k would remain the same and could not be a robust evaluation. This limitation is tied to the nature of Precision@k as a metric, which is only focused on the top 'k' recommendations and does not consider the full ranking of recommended items. Possible solutions to this specific limitation are proposed, one of them is utilizing the NDCG (Normalized Discounted Cumulative Gain) as evaluation metric. The NDCG accounts for the relevance of an item as well as its position in the list of recommendations. NDCG assigns higher scores to relevant items that appear earlier in the list, so even if on a setting that has a fixed “k” number of relevant items within “n” recommendations, the model would still be encouraged to rank them as high as possible. However, this evaluation metric takes the rankings of the products in consideration of relevance and this data is absent in our case, which was the main reason for focusing on precision@k in the first place.

In order to confirm whether the limitation of Precision@k applies to our system or not could be proved by running diverse tests which the tests run and the discussion are given in chapter 4.

4.Implementation

In this chapter, following the design decisions explained in the chapter 3, we explain the implementation of the system following these steps in detail in their respective sections:

- Setting the project environment
- Data preparation
    - Creation of the database
    - Parsing, formatting the data and handling missing and anomalous values
    - Population of the data
    - Creation of category table and the hierarchy between the categories
    - Creation of foreign key links between the tables, including additional columns
    - Retrieval of the data, creation of user-category matrices per level.
    - Normalization of the data
- Implementation
    - Creation of user profiles
    - Creation of users' taste space
    - Assigning the users to their respective taste space
    - Recommending Products
    - Evaluation of Recommendations

4.1.Setting the Project Environment

We created the project using the Java programming language version 17 and used Apache Maven as a build tool for the ease it provides in package and dependency management and included the following libraries:

Library Name	Purpose
MySQL Java Connector	To establish the database connection
Apache Commons CSV	To read the csv files in a fast manner
Weka Tool	To use the unsupervised clustering algorithm (KMeans)
Apache Mahout	To apply collaborative filtering: -Calculation of pearson correlation -To use the instance based learning algorithm (KNN)

Table 4. The libraries used and the purpose of use

For the MySQL server we used Xampp Server in our local setting and connected to this local server through our project.

After these first settings were done, we moved forward to the data preparation step.

4.2.Data Preparation

In this section we aim to explain the data preparation part in detail. Data preparation and preprocessing steps are indeed crucial steps while working with the real data, as the anomalies, outliers or the dangling entries on the data yields misleading results. As a natural result of working with the real data, the preparation of the database has taken quite an important part of the time dedicated to this Master Thesis. The data preparation step usually does not deserve much attention in books, but when dealing with real application, it raises many challenges, as we have experienced in this real study. After overcoming many technical challenges of the real data, and achieving to have a program that creates the database with a time of about 7 hours, we moved to the data preprocessing stage.

4.2.1.Creation of the Database and Parsing, Formatting and Population of the Data

4.2.1.1.Creation of the Database

In order to store, retrieve and easily filter and study the data we have created a MySQL database with a better organization of the tables in order to optimize the creation of the user-category matrix. The database consists of five tables. The entity-relationship design of the database and the format of the data to be stored are given in the design section.

When we began to populate the data into the database, there was a vast amount of data which MySQL was not able to import from the csv files directly, so we were forced to write specific Java methods to manage the manipulation, transformation and preparation of the data.

4.2.1.2.Parsing, Formatting the Data and Handling Missing and Anomalous Values

The data was read from the csv files line by line, splitted by the semicolons, parsed into their respective data types as shown in Table 1 and inserted into the corresponding table's columns in the java program. While parsing the real data, the first one of the many challenges we have faced was the data type. The data obtained seemed to be integers, strings… However, there were also so many mixed types of entries in the same column. The screenshot below is an instance from the clientes.csv (clients) file where the postal code was first set as varchar. This clearly was not a correct entry. In order to eliminate this, we have discarded all the entries that were not parseable into integers.

codigoPostal	anoNacimiento
PEP	1966

Figure 7. Example of an incorrect, unparsable to integer client entry.

Continuing with the client entries, there were many entries of clients which contained only the id, there was no postal code or year of birth available as seen in the screenshot below.

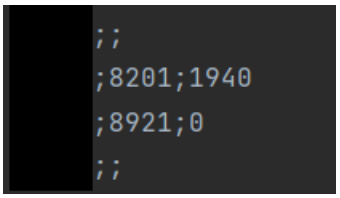

Figure 8. Examples of incomplete client entries.

We intended to build the user clusters based on their demographics to study the change in recommendations based on user demography and due to this reason, we deleted the rows that were missing these data. We also deleted the anomalous values such as year of birth entered as 0,which an example of it can be seen in the screenshot above.

Though it was hard for us humans to see among 16656306 lines of order detail entries, there were inconsistencies of date separators which programmatically caused format exceptions since some entries had the date as 01-01-20 and some as 01/01/20. The time entries were given as 11:26:03 AM which is not supported by MySQL. MySQL timestamps are stored in 24-hour format. We parsed each date and time entry and formatted them in order to have a single format and wrote them to the database after this formatting phase.

4.2.1.3.Population of the data

After parsing, formatting and handling the missing or anomalous data, we wrote the obtained clean data, to our database. Writing such vast amounts of data even while utilizing the connection pooling strategy took around 7 hours to complete for all the tables. Along with the technical challenges of working with real data, another big and vital challenge was the read and write times required.

4.2.1.4.Creation of Category Table and the Hierarchy Between the Categories

In order to preserve the hierarchy between the categories while writing them to the categories table some dedicated functions have also been implemented. The hierarchical category information were given in each line of the article file, which are of the form:

artileID; Redondo atado de ternera; Alimentación > Carnes y huevos > Ternera

Parsing the category information by utilizing the separator of “>” between them we were able detect the position of the category and by using this position and considering that the structure of the categories are the same for all the entries of products we assigned the categories their extracted respective level and wrote them to the database. For the entries in the example the ids are assigned (automatically) and the entries are written as follows:

Category	Assigned Id	Level
Alimentación	1	0
Carnes y Huevos	2	1
Ternera	3	2

Table 5. An example of how the hierarchy between categories are preserved

In order to establish the parent-child relationship, for the entries that had the level 0, we inserted their parents as NULL and for the rest of the entries, for example “Carnes y Huevos” we searched for the id of “Alimentación” entry and inserted its ID as parent ID. The following table shows the final representation of the example entries in the categories table:

Category	CategoryId	Level	Parent Id
Alimentación	1	0	NULL
Carnes y Huevos	2	1	1
Ternera	3	2	2

Table 6. An example of how the categories are stored

After all the entries were written we set the parentId column to be foreign key pointing to the categoryId column inside the same table which enabled us to link the current category and their parents with their ids in a recursive way and the indexing

done on this process gave us the benefit of fast access to the upper levels of categories as well as filtering the categories by specifying their levels or their parents.

The screenshot in Figure 4 shows a section from the categories table we created. The ids were automatically generated. We can observe that only by hovering on the id number 4 in the parentCategoryId column, it tells us the parent category (Bebidas) of the category Licores.

categoryId ▲ 1	categoryName	parentCategoryId	level
1	Alimentación	*NULL*	0
2	Carnes y huevos	1	1
3	Ternera	2	2
4	Bebidas	*NULL*	0
5	Alcohólicas de alta graduación	4	1
6	Licores	4	2
7	Droguería	*NULL*	0
8	Productos de un solo uso	7	1
9	Bolsas de basura	8	2

Bebidas

Figure 9. A section from the categories table

4.2.1.5.Creation of Foreign Key Links Between the Tables, Including Additional Columns

After writing all the data to their respective tables, in order to establish the foreign key links between the lineaspedidos (order details) and pedidos (orders) table we needed to delete all the entries that did not have a respective order information in the pedidos (order) table. The following screenshot shows this deletion process and the number of rows affected.

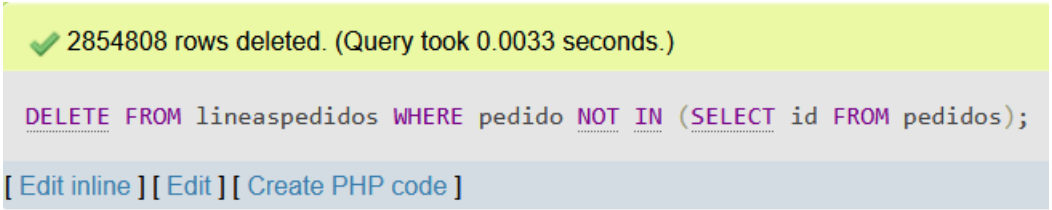

Figure 10. The query executed to delete the dangling entries

Then we included the client column to the lineaspedidos (order details) table by joining it with the pedidos(order) table in order to obtain the respective client ids and populated the new column with these ids. After the population ended we linked the tables with the foreign keys. The foreign key links among all the tables are given in the proposed database schema in the design section.

4.2.1.6.Retrieval of the data, Creation of User-Category Matrices per Level

Retrieval of the data as planned, started from the lineaspedidos (order details) table and by the retrieval of the articles purchased by the clients along with the quantity of the purchase we were able to hold all of this information in map data structure in our program. Then, by retrieving the categories per level and matching the articles with their respective categories we were able to build the user-category matrices of different levels of category and fill them with the total quantity of all the purchases made per category.

It is important to point out that the required time span for retrieval of all entries from the database and creation of the user-category matrix per level on average takes around 2 hours. This required time is not subject only to the number of entries to be read but also to the recursive calls made in order to detect the desired category of an article given a level. That is, if we have an article of milk in the milk level (which corresponds to level 2) and we wish to create a user-category matrix of level 0, we need to make 2 calls in order to find the parent category with the level 0 which corresponds to Alimentación (food). Even though indexing the database using foreign key links between the child-parent category pairs allows for rapid access, the sheer volume of entries - 12,480,997 to be precise - that need to be processed to create the user-category matrix negates this advantage. In order to avoid spending 2 hours on average to build the matrices everytime the program runs, we benefited from working in a static environment and after building the matrices we saved them as Java object serializable files. This approach indeed gave us the ability of running the program as many times as we wanted without requiring us to wait for the matrices to be created from scratch. The screenshot below shows us the read time required in order to read the user-category matrix of level 0 that was stored as a Java serializable object is 1240 milliseconds.

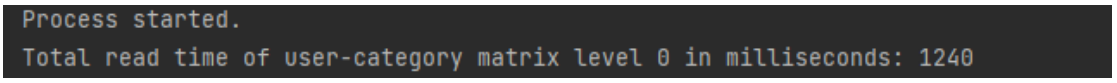

Figure 11. A screenshot showing the read time required

4.2.1.7.Normalization of the Data

After overcoming many obstacles related to the data, the read or processing times we were able to work on the user-category matrices. By utilizing the respective function provided in WEKA library, we applied the min-max normalization on the

user-category matrices in order to avoid the outliers in the data to yield misleading results.

4.3.Implementation of the recommender system

In this section, we explain the implementation of the recommender system by utilizing the adequate methods of Artificial Intelligence. The roadmap of this stage as given in the beginning of this chapter, determined as follows:

- Creation of user profiles
- Creation of users' taste space
- Given a user ID, finding the similar users
- Recommending Products
- Evaluation of Recommendations

4.3.1.Creation of User Profiles

In order to create the user profiles we applied an unsupervised clustering algorithm to the user-category interaction matrix. Our preferred algorithm was KMeans as explained in the design section and using the related function provided by the WEKA library we obtained the clusters of users which correspond to the user profiles based on their purchase patterns. Finding the optimal number of clusters is indeed an important point of working with the KMeans algorithm. In order to find it, we ran several tests on the 3 user-category matrices we have created with different numbers of K and calculated the SSBetween / SSWithin ratios per test and compared the results. The detailed results of these tests are given in the results section however we can mention here that at the point where there was a sign that the data was starting to overfit, we decided the number of optimal K.

4.3.2.Creation of Users' Taste Space

By obtaining the user profiles based on their purchase patterns, we could proceed to extract the "taste space of the users who share similar purchase patterns". In order to create the taste spaces, we applied collaborative filtering since the taste space is determined by taking the collective ideas of the users into account. In order to do so, within each cluster of users, we calculated the pairwise similarities of users by utilizing the Pearson correlation similarity metric and the adequate function of Apache Mahout that does the calculation using this similarity metric. Since the pairwise calculation of Pearson correlation was made for all the users in their respective clusters and the outcomes are stored in the respective data structure of Apache Mahout, during the testing phase we were able to see the similarity measures of two users (given their user IDs) that have the same purchasing patterns. In fact, the high similarity between the users demonstrated to us that they share the same "taste space".

4.3.3.Assigning the Users to Their Respective Taste Space

After the “taste spaces” of the users are found, the next step was to assign the users to their respective taste spaces. In other words, grouping the users that share the most similar preferences. In order to do so, by utilizing the built in data structure of Apache Mahout which was holding the results of similarities among all the users within a cluster and the adequate function the same library provides for running the K Nearest Neighbor algorithm, we found the groups of most similar users and stored them in a map data structure.

As we explained in the design section, due to the vast amount of data there is a high possibility of some users not having so many similar users to them. In other words, there is a high possibility for the taste spaces to be sparse. Though the K Nearest Neighbor algorithm is efficient as the data evolves, in a static setting like ours in order to get robust results we had to find and set the number of neighbors to be calculated (K) to an optimal number. Another technical reason for finding the optimal K was that Apache Mahout was returning an empty result set, when the set K for example, is set to six and there are no six neighbors available. We ran several tests in order to find the optimized number and we detected the optimized number of K that gave us more robust results. The details of the tests ran are given in the results section.

4.3.4.Recommending Products

When we detected the users that shared similar purchase patterns and similar taste, we could start recommending products. Our recommended articles list was designed to have:

- A portion of articles based on the past purchases of the user
- A portion of articles based on the past purchased of the similar users

For this stage, we created a function that accepts a user ID for the sake of simplicity, in this explanation we will call this user A; for the portion of articles/products to be recommended based on the past purchases of the user we retrieved random products simply by using the user ID given and we will call these articles portion 1 of recommendation list.

For the portion of the articles/products based on the purchases of the similar uses; from the “taste space data” we hold, our function retrieves the users who had the similar preferences to the user A we can call these similar users B and C. By retrieving the distinct articles/products purchased by the users B and C we obtain the portion 2 of the recommendation list.

By randomly selecting the articles from both of these portions and merging them, we obtain a recommendation list of products. The randomized selection of articles from both portions to create the final recommendation list ensures diversity, which can enhance the user's discovery of new products they might be interested in.

For the case where the fixed number of recommendations can not be provided from these two portions of articles, we created a functionality that by utilizing recommended articles of similar users B and C, we find the innermost level of categories of these articles / products. And by utilizing these category information, we retrieve random products within the same category in order to fill the recommendation list. The results of the recommendation phase are given in the results section.

4.3.5.Evaluation of Recommendations

In this section we explain how we implemented the evaluation function. As we pointed out in the design section, due to the absence of product ratings our preferred evaluation metric was Precision@k and it calculates the proportion of recommended items in the top-k set that are considered to be relevant. A high precision@k means that most of the products recommended to the users are indeed relevant. Relevancy in our case is considered to be the articles / products recommended that correspond to products already bought by the user in the past purchases. We implemented the functionality that takes the relevant item list and the total recommendation list and calculates the proportion of them. However, as we explained in the design section, the precision@k suffers from evaluation heavily depending on “k” and, in our case, the introduction of additional products from random category-based selection also hampers this quality measure.

The addition of non-purchased products in the recommendation list is a technique used with the aim of diversifying the suggestions, as well as to introduce some serendipity in the list. Making the customer discover different products available in the same categories that he/she buys is a way of engaging him/her to buy more. If the client decides to buy one of these new products, on the one hand, he/she may discover something that may like him/her, and on the other hand, permits the shop to sell more products.

Another consideration that may be taken into account during evaluation of the quality of the recommendation is the possibility of a re-definition of the concept of “relevance”. Although in a supermarket context our preferences do not change rapidly, in order to overcome the limitations of the precision@k metrics, a score of relevance could be given differently to the products recommended based on the user’s past purchases taking into account time considerations. The metric of relevance could include the recency of the purchase date or the purchase frequency of the respective product in the past. This method could change the definition of relevancy and applying these metrics in order to detect the most relevant articles / products could yield good evaluation results. However, there could be possible new biases of articles purchased more frequently, such as water or bread, having the highest relevance score and dominating the evaluation results. We leave the evaluation of the results based on a metric in the absence of product ratings as open to discussion.

5.Results

In this chapter, we explain the results obtained throughout the implementation of the system by following this roadmap:

- Data preparation
    - Results from data cleaning: Handling missing, anomalous, and dangling data
    - Results of data extraction and normalization: creation user-category matrix
- Implementation of recommender system
    - Results of creating user profiles
    - Results of creating user's taste space
    - Results of assigning the users to their respective taste space
    - Results of recommendation
    - Results of evaluation of recommendations

5.1.Data Preparation

5.1.1.Results From Data Cleaning: Handling Missing, Anomalous, and Dangling data

For handling the missing, anomalous and dangling data, we have deleted them. In the table below we can observe the number of entries in each database table before and after all the data preprocessing steps applied; so, we can see how applying the data preprocessing steps influenced the overall dataset size.

Table Name	Number of Entries Before	Number of Entries After
lineaspedidos	16656305	12480997
pedidos	564612	460184
clientes	62590	46174
articulos	22009	21662

Table 7. The effects of data cleaning on the dataset

- We can observe that almost 16 thousand users are not used to make the recommendations to be made by our system.
- 347 articles will not be utilized for recommendation due to missing category data.
- 4.175.308 order details can not be utilized to create the user-category interaction matrices due to missing data. As the number is huge, the absence or existence of this amount of data could yield different taste spaces to be created.

5.1.2.Results of data extraction and normalization: creation user-category matrix

After the data preprocessing steps were complete, we moved to create the user-category interaction matrix. The successful creation of this matrix, without any exceptions or errors, served as a practical confirmation that our data was in a suitable state for further analysis. Consequently, we then proceeded to extract the data about the users and the specific articles they had purchased. From these articles, we retrieved the categories they reside in the 0th, 1st and 2nd level. So for the article “Leche entera” was represented as “Leche” in the 2nd level, as “Lácteos y derivados” in the 1st level and as “Alimentación” in the 0th level. The figure shows us a portion from the user-category interaction matrix for level 0 of categories with the first column being an identifier of the user and rest of the columns correspond to the categories, whose value indicates the number of times the user purchased a product of the corresponding category (i.e. number of interaction).

```
3,1,4,2,1,1,1,1,1,0,0,1,1,0,0,0,1,0,1,1,0,3,0,0,0,0,0,0,0
4,4,0,2,0,0,0,2,2,0,0,1,0,0,0,0,0,0,0,2,0,0,0,0,0,0,0,0,0
5,1,1,1,0,1,1,2,1,0,1,1,1,0,0,0,1,0,1,1,0,0,0,0,0,0,0,0,0
6,0,0,0,0,0,8,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
7,1,4,1,2,1,0,1,1,0,0,1,2,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0
8,1,1,1,0,1,0,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
9,1,1,1,1,1,1,1,1,1,0,0,0,2,0,0,0,2,0,1,3,0,0,0,0,0,0,0,0
11,4,0,0,0,0,0,0,2,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
12,2,1,1,1,0,0,1,1,1,0,0,1,0,0,0,1,0,0,2,0,0,0,0,0,0,0,0,0
13,1,6,2,1,1,1,1,2,1,0,1,4,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0
16,1,1,2,1,2,2,1,1,0,0,1,1,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0
17,1,6,2,4,1,3,6,1,0,0,4,4,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0
20,1,1,1,0,1,0,2,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
21,4,1,1,1,1,10,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
22,3,2,1,1,1,0,1,2,0,2,0,0,0,0,0,0,0,6,1,0,0,0,0,0,0,0,0,0
23,1,6,1,0,1,0,0,1,0,0,2,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0
24,1,1,1,0,0,1,2,1,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
25,1,12,1,0,0,8,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
26,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
28,1,2,1,0,6,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
```

Figure 12. A screenshot from a portion of user-category matrix

We can observe that, in general, the number of purchases are mostly below 5 in this example. The data matrix is also quite sparse, as users tend to focus on buying products of a small number of categories.

We can also see that some users buy few products, like user 28; while others buy more, like user 17, who bought 33 products (probably on different days).

In this phase we have obtained 3 user category matrices by utilizing different levels of categories. The dimensions of these matrices are given in the table below.

Dimensions (users x categories)	Category Level
46174 x 23	0
46174 x 127	1
46174 x 412	2

Table 8. Dimensions of the user-category matrices by their respective category levels

5.2.Implementation of recommender system

5.2.1.Results of creating user profiles

By the application of an unsupervised machine learning algorithm, the KMeans, we could obtain the user profiles. As we explained in the previous sections, finding the optimized number of "K" is a crucial task in order to avoid the model overfitting the data. We have run several tests with different numbers of K and evaluated the score by using a ratio of SSBetween / SSWithin. The table xxxx is a portion from the results table we have prepared to compare the SSB/SSW ratios that were calculated for 25 different numbers of clusters on the user-category matrix of category level 0.

Number of clusters (k)	SSB	SSW	Ratio (SSB/SSW)
3	1219857.887416967	1953382.5352292887	0.6244848950048484
5	1397876.23092072	1775364.19172194	0.7873743525067431
10	2008096.4816000804	1165143.94103904	1.723475023875008
15	2120172.9561559837	1053067.4664858354	2.0133306019140056
20	2141278.6912955623	1031961.7313452474	2.0749593965119506
25	2354805.3309762003	818435.0916635821	2.8772047471592814
30	2496904.6285103294	676335.7941304831	3.691812040970599
31	2480822.44862059	692417.9740204052	3.582839472257087

32	2476228.763423109	697011.659218043	3.5526360724024584
33	2473889.795636438	699350.627004596	3.5374098486654817

Table 8.A portion of calculated SSB, SSW and ratios per given K

We can observe that the SSB/SSW ratio increases as more clusters are added (indicating that the clusters are becoming more distinct from each other), but it starts to decrease at k=32. This was a sign that the data was starting to overfit.

We had a similar view of the ratio being decreased when the number of clusters were changing from K=30 to K=31 on all matrices. Therefore, we have chosen K=31 as a good balance point, as at this point we obtained a relatively high SSB/SSW ratio and less number of clusters, which makes the model more interpretable. With K set to 31 we applied the clustering. Figure xx shows a portion from the program output expressing the assigned clusters for the first 10 instances of the user category matrix.

```
KMeansWeka | Running KMeans for clusters: 31
Number of rows: 46174
Number of columns: 23
Instance 1 is assigned to cluster: 10
Instance 2 is assigned to cluster: 20
Instance 3 is assigned to cluster: 16
Instance 4 is assigned to cluster: 1
Instance 5 is assigned to cluster: 10
Instance 6 is assigned to cluster: 19
Instance 7 is assigned to cluster: 10
Instance 8 is assigned to cluster: 9
Instance 9 is assigned to cluster: 30
Instance 10 is assigned to cluster: 26
```

Figure 12. A screenshot demonstrating the assigned clusters of users

5.2.2.Results of Creating User's Taste Space

The taste space of the users was created by applying the collaborative filtering by means of calculating the Pearson correlation similarity on each user that is in the same cluster. This pairwise comparison enabled us to detect the users who shared a similar taste. In the screenshots given below, we compared the users with IDs 24, 39, 716 and 1518 who were in the cluster number 0. We can observe that the users with IDs 24 and 39 are the most similar ones, whereas out of these three examples the users 24 and 716 are the least similar ones. Thus it would be correct to say that out of users with IDs 24,39, 716 and 1518; 24 and 39 share the same taste space.

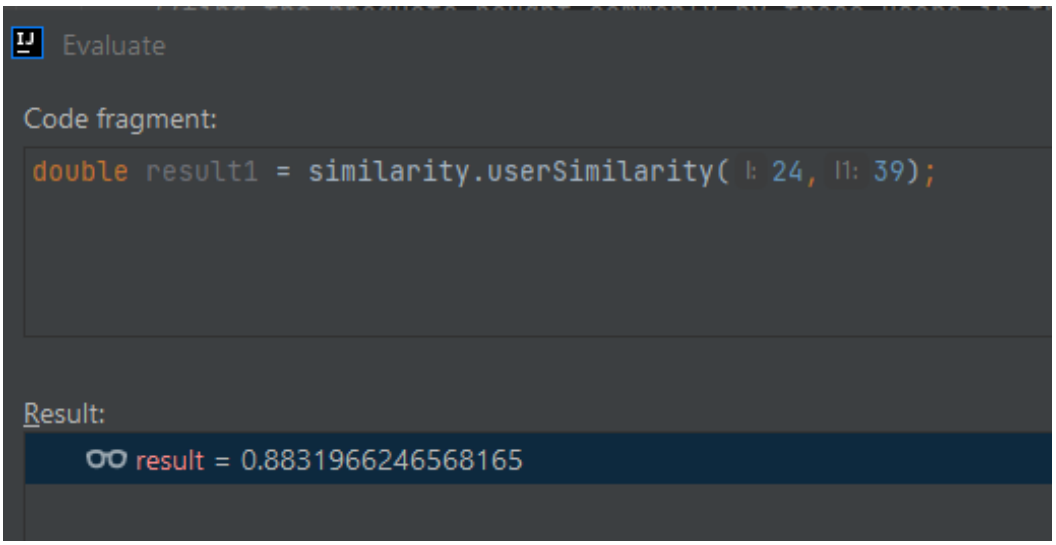

Figure 13

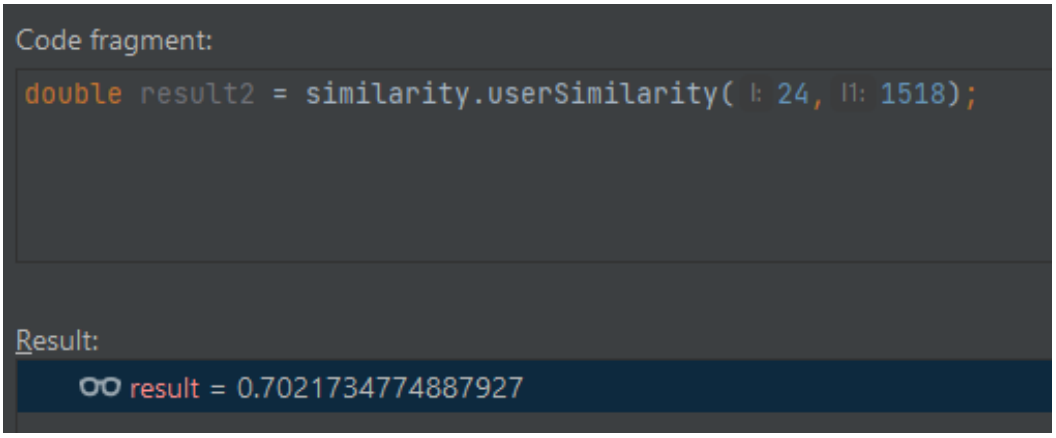

Figure 14

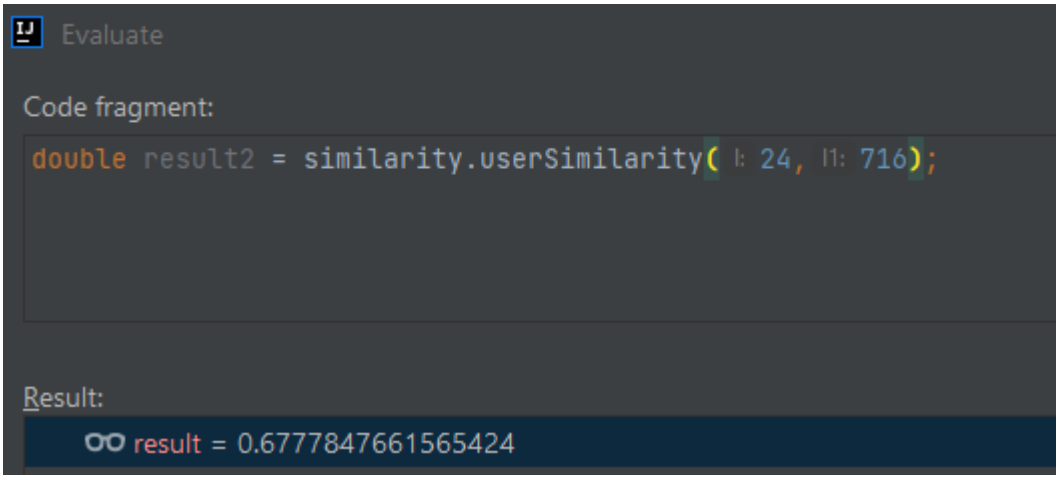

Figure 15

Figures 13,14,15 screenshots demonstrating the similarities between some users

5.2.3.Results of Assigning the Users to Their Respective Taste Space

Since the high similarity between two users represents their shared taste space, by utilizing the K Nearest Neighbor algorithm we aim to find the “neighbors” of shared taste spaces.

Even though the K Nearest Neighbor algorithm is capable of finding the optimal number of neighbors; in our static setting, we focused on determining the most optimal number of neighbors to be retrieved. Given the large number of users, setting the algorithm to retrieve the top five or six similar users often resulted in Apache Mahout returning an empty array due to the high possibility of many users not having the five or six closest neighbors. Therefore, we conducted several tests to optimize this parameter and generally obtained more robust results when retrieving the top three similar users. This decision was driven by the need for a balance between maximizing the number of similar users and ensuring the reliability and consistency of the results.

In the given screenshot below we can see that for the user with ID 24, the closest three neighbors are found to be the users with the IDs 2187,10711 and 12118. This signifies that these three users are sharing the same taste space and anytime one of them asks for a recommendation the preferences of the other three(user IDs 24, 2187, 10711, and 12118) will be considered.

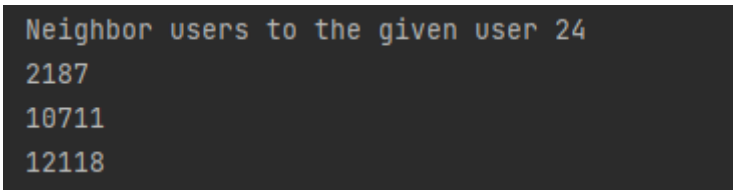

Figure 16, a screenshot showing the three most similar users to the user with ID 24

5.2.4.Results of Recommendation

In this section, we would like to give some screenshots of the recommendations generated by our program. The figures 17 and 18 are the recommendations generated by using the level 0 of the user-category matrix.

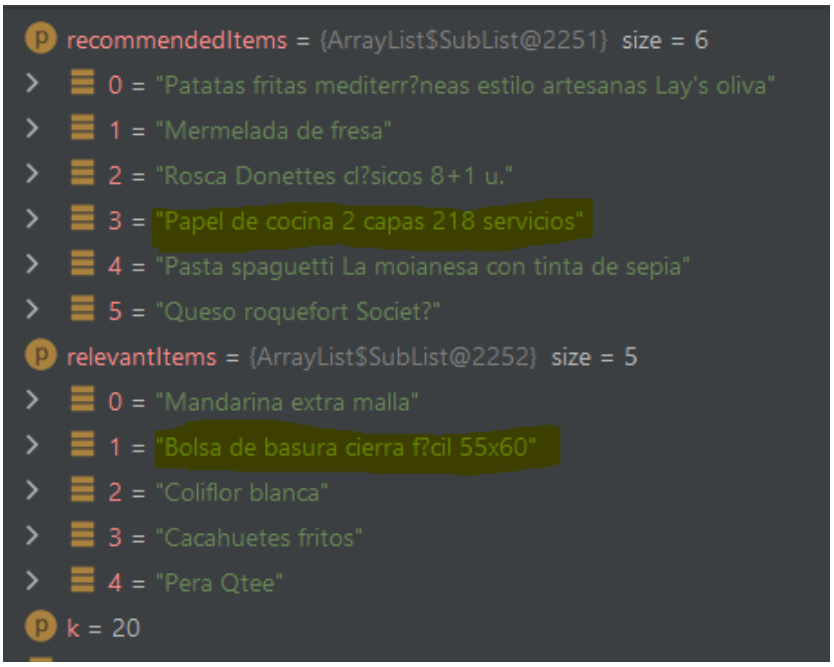

Figure 17, screenshot of recommendations made for level 0 of categories

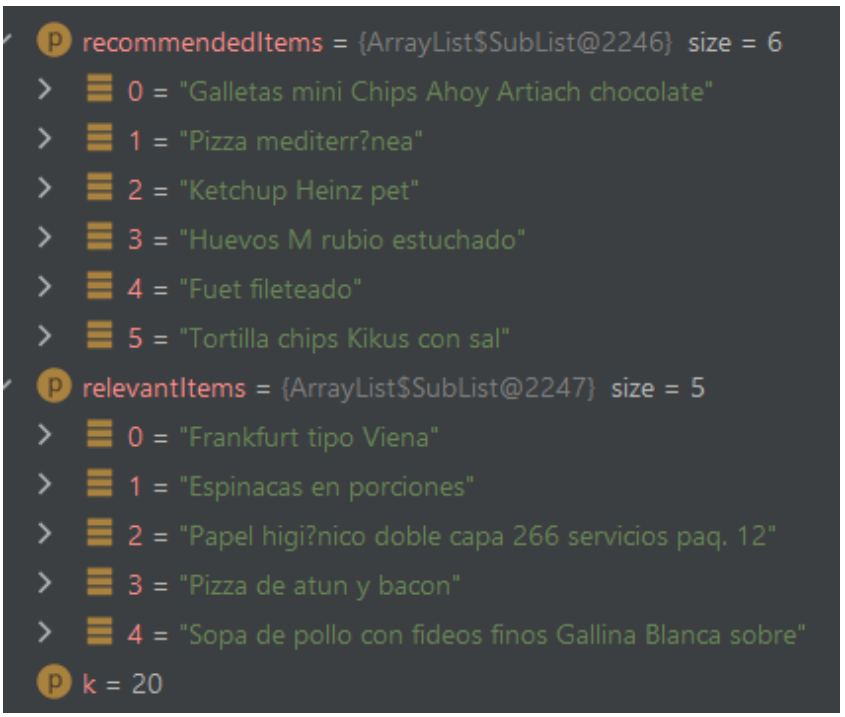

Figure 18, screenshot of recommendations made for level 0 of categories

The figures 19 and 20 are the recommendations generated by using the level 1 of the user-category matrix.

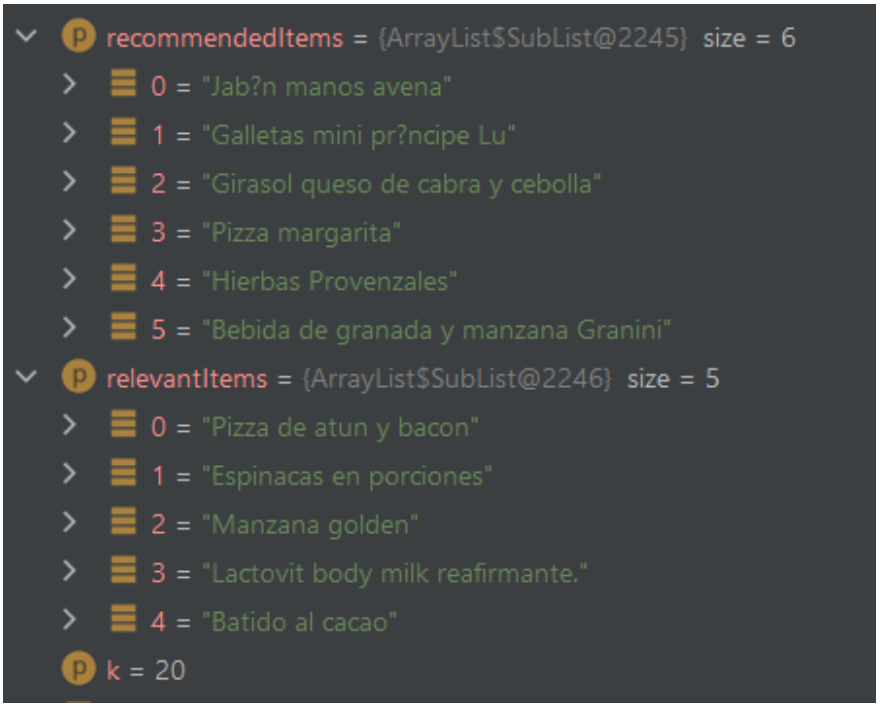

Figure 19, screenshot of recommendations made for level 1 of categories

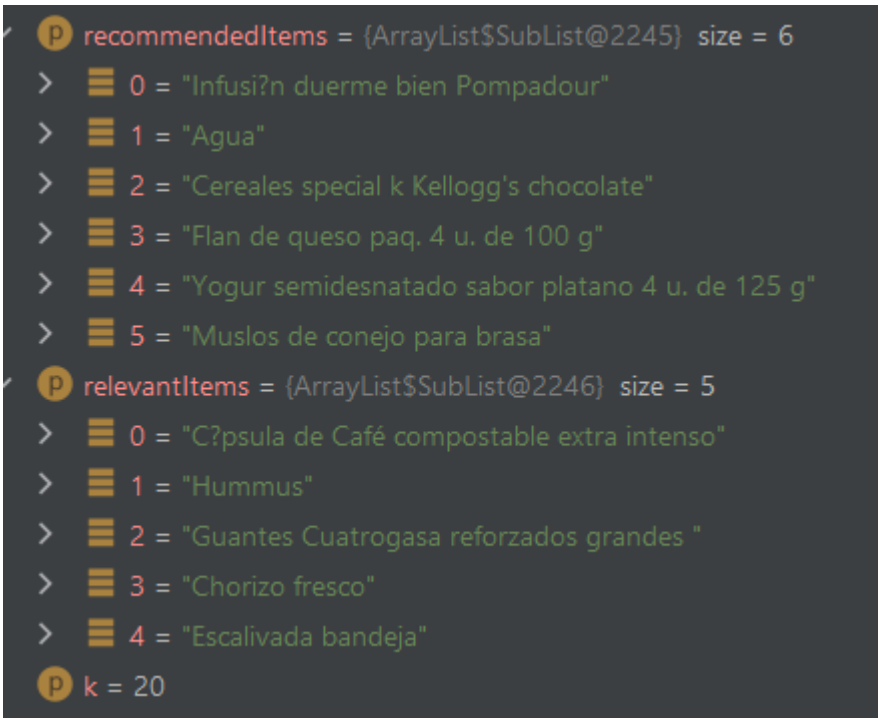

Figure 20, screenshot of recommendations made for level 1 of categories

The figures 21 and 22 are the recommendations generated by using the level 2 of the user-category matrix.

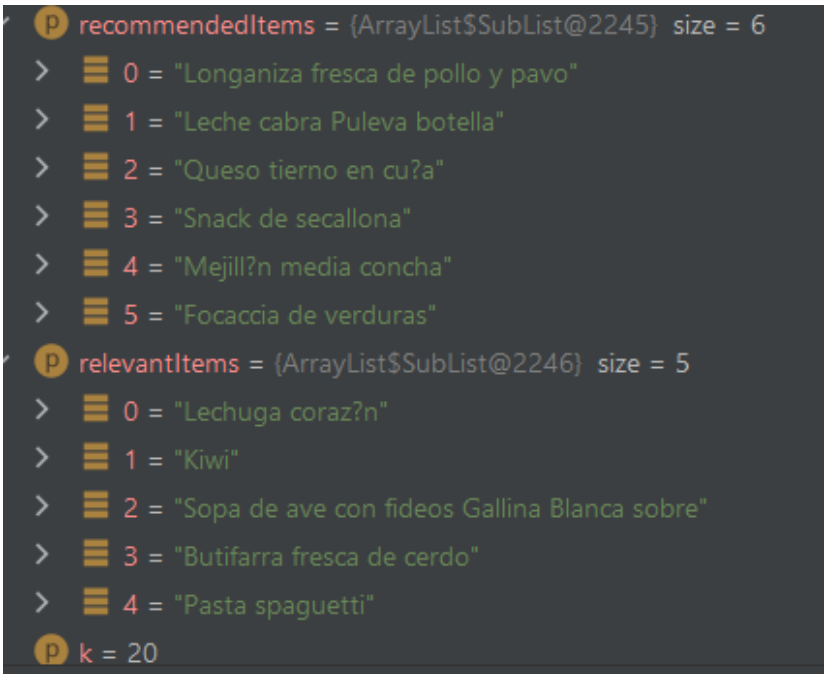

Figure 21, screenshot of recommendations made for level 2 of categories

Figure 22, screenshot of recommendations made for level 2 of categories

We can observe from the results that the absence of any criteria - such as product characteristics, relevance of purchase data, ratings, or any other factors that could be used to establish a network or relationship between products - hampers our ability to provide accurate recommendations. Moreover, we observe that although utilizing categories at a higher level prevents the recommendation of excessively diverse products, it still lacks the refinement needed for optimal performance.

6.Conclusions

This research has presented an in-depth exploration into the realm of recommendation systems, specifically focusing on designing and implementing a system that takes into account user behaviors and preferences.

We presented the methods available in the literature, and we have studied how they can be used for product recommendation for an online food supply chain. With the data provided by the company, a suitable recommendation model has been proposed.

Throughout the design, implementation, and testing chapters, we emphasized the inherent challenges of working with real-world data during the data preparation phase and presented our approaches to overcoming most of these challenges. We explained our approach for working with sparse data and the optimizations we have done in order to speed up the time consuming processes.

After the data pre-processing, we highlighted the utility of using collaborative filtering in the process of creating user taste spaces and assigning users to their respective spaces. A clustering technique has been used first to find a set of general profiles. Second, neighbors can be found in the clusters by means of similarity calculation between the user's previous purchases. A list of products is obtained from similar users and they are completed with a set of random products extracted from the same categories.

Lastly, we highlighted the intricacies of working with large and diverse datasets. We pointed out the limitations caused by the absence of criteria to establish relationships among articles or products and discussed the impact of these constraints on various stages of the implementation process.

As future work, we suggest extraction of any criteria suitable in order to link the products as this is a crucial part of recommendation. It would also be important to test the recommendations with real users and have their feedback in accuracy.

References

[1] Khodabandehlou, S. (2019). Designing an e-commerce recommender system based on collaborative filtering using a data mining approach. *International Journal of Business Information Systems*, *31*(4), 455-478.

[2] Schafer, J. B., Frankowski, D., Herlocker, J., & Sen, S. (2007). Collaborative filtering recommender systems. *The adaptive web: methods and strategies of web personalization*, 291-324.

[3] Hu, Y., Koren, Y., & Volinsky, C. (2008, December). Collaborative filtering for implicit feedback datasets. In *2008 Eighth IEEE international conference on data mining* (pp. 263-272).

[4] Lee, J., Sun, M., & Lebanon, G. (2012). A comparative study of collaborative filtering algorithms. *arXiv preprint arXiv:1205.3193*.

[5] Bhatta, R., Ezeife, C.I., Butt, M.N. “Mining Sequential Patterns of Historical Purchases for E-commerce Recommendation”. In: Ordonez, C., Song, IY., Anderst-Kotsis, G., Tjoa, A., Khalil, I. (eds) Big Data Analytics and Knowledge Discovery. DaWaK 2019. Lecture Notes in Computer Science (2019) vol 11708, 57–72.

[6] Su, Xiaoyuan, and Taghi M. Khoshgoftaar. "A survey of collaborative filtering techniques." *Advances in artificial intelligence* 2009 (2009).