Recommendation system for supermarket online shopping

Master Thesis
submitted to the Faculty of the
Escola Tècnica d'Enginyeria de Telecomunicació de Barcelona
Universitat Politècnica de Catalunya
by

Cai Badal i Regàs

In partial fulfillment
of the requirements for the master in
Master's degree in Advanced Telecommunication Technologies Engineering

Host university advisor: Stephen Ho
Home university advisor: Montse Pardàs
Barcelona, 1^{st} Sept. 2022

Contents

List of Figures

Revision history and approval record

Revision	**Date**	**Purpose**
0	01/06/2022	Document creation
1	01/07/2022	Document revision by host supervisor
1	03/07/2022	Document revision by home supervisor
2	05/08/2022	Document revision by host supervisor

DOCUMENT DISTRIBUTION LIST

Name	**e-mail**
Cai Badal	cai.badal@estudiantat.upc.edu
Montse Pardàs	montse.pardas@upc.edu
Stephen Ho	ssh1@mit.edu

Written by:		Reviewed and approved by:	
Date	01/09/2022	Date	01/09/2022
Name	Cai Badal	Name	Stephen Ho
Position	Project Author	Position	Project Supervisor

Abstract

With the recent rise of online purchasing, many companies have focused on developing recommendation systems, where customers are suggested different options for complementing their purchases. This thesis will introduce and test four different approaches for a recommendation system for online shopping at supermarkets, based on the historical of previous customers. The four proposed algorithms are based on successful recommendation systems, which include a histogram-based approach, a graph theory-based approach, an embedding-based approach and finally a support-vector machine-based approach.

1 Introduction

Recent years have shown a significant growth of the buying and selling of manifold possibilities of goods and services over electronic networks, mainly Internet, all around the world. These online processes are known as E-commerce [1]. This trend is continuing to grow with the development of new technologies and is predicted to keep increasing in the following years [2]. The following figure shows this shifting of conventional purchases into e-commerce, with an outstanding growth of worldwide online purchasing.

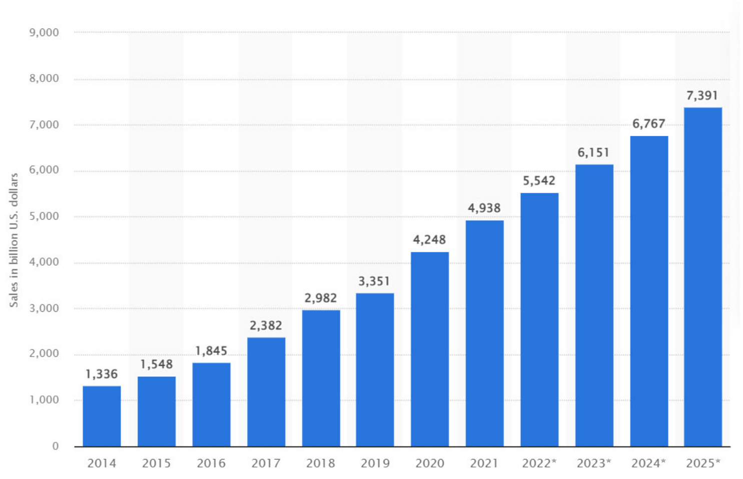


Figure 1: E-commerce sales worldwide from 2014 to 2025.[2]

With this technological expansion, many retailers have chosen to offer their customers online service, giving them the option to purchase different goods via internet. According to a survey performed to 1000 UK customers, 91% of the customers find either *Important* or *Very important* the user experience while online shopping [3]. Nonetheless, one of the main downsides of E-commerce is the lack of a face-to-face expert or employee to help with decisions on purchases.

Both the enormous growth of variety on data on the e-commerce sector and the lack of this human support might lead the customers into ill-advised shopping decisions, and many businesses find critical to offer shoppers some guidance and relevant information to assure customers can make appropriate choices. Companies have opted to develop systems which can help customers to take decisions based on individual information of the clients, systems which are known as Recommendation Systems (RSs).

Lidl is a German international retailer chain, with over 10 thousand stores operating in 27 different European countries. In 2017 they expanded to the United States, with more than 150 active stores [4]. This project will focus on developing a Recommendation System for the online purchases for *Lidl*, based on the historical buys in some of their stores. These recommendations will be purely based on what is contained in the customer shopping list, suggesting a new item available in *Lidl* stores as an addition to what is going to be purchased. The dataset used is a history of previous purchases in *Lidl* supermarkets from which recommendations will be obtained.

The thesis will be divided into four sections. Following this introduction, some state of the art approaches for recommendations will be introduced, followed by the explanation of four different tested approaches for this use case. After that, the results of the experiments will be introduced. Finally, the thesis insights will be summarized in the conclusion, including some future steps.

1.1 Gantt Diagram

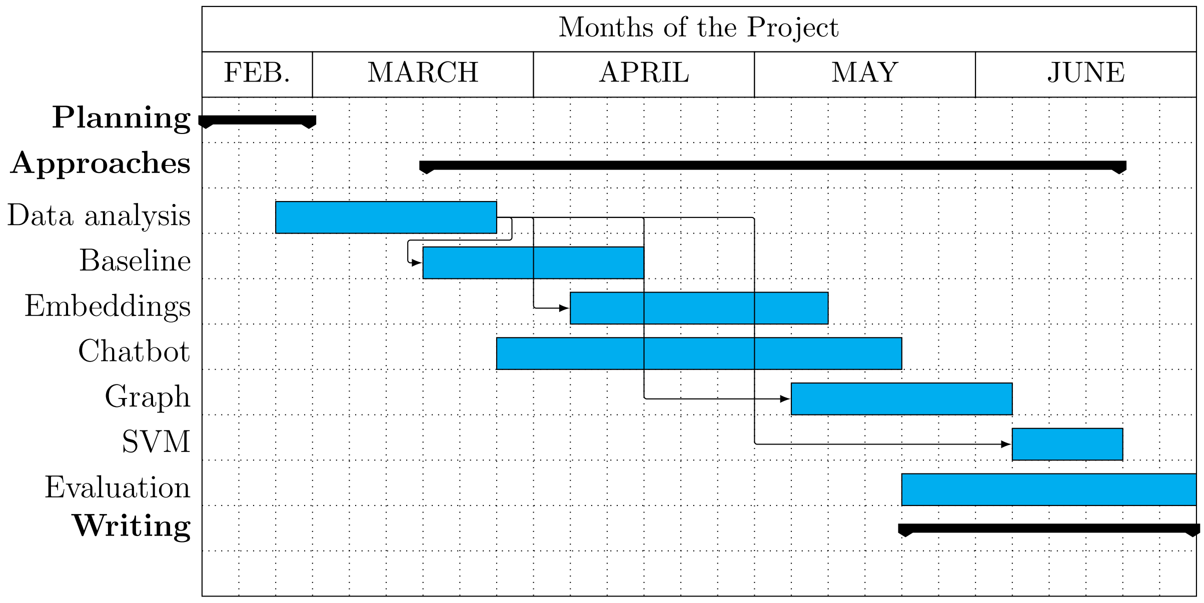


Figure 2: Gantt diagram of the project

The previous figure shows the distribution of the work during the development of the thesis. After some planning, and setting up the environment at the MIT Auto-ID Lab, the first phase of the project started with an analysis of the available data and the exploration of existing recommendation systems that would fit into this use case. The following weeks were focused on each of the proposed approaches that will be described in the succeeding sections. The process of this thesis ended with the evaluation of each of the approaches and the writing of the final document.

As it can be seen, a very important part of the project was the *Chatbot* development. In order to offer the *Lidl* users a better experience for the online shopping, a baseline chatbot was built. With this chat, users could interact directly with a computer program that was able to guide them through the buying process, and give them suggestions depending on what was contained on the shopping list.

The development of this initial chat-bot was done jointly with another student at the lab. During the initial phases of this bot, the conversation was completely rule-based, and did not involve extensive research or investigation. Therefore, the work that has been done regarding the chat-bot will be presented briefly in the appendices section.

2 State of the art of Recommendation Systems

Recommendation Systems (RSs) are all the set of techniques that can use meaningful information from the customer to provide some suggestions as guidance. This project focuses on suggestions to complete the shopping list in a supermarket, with a complementary item or product. Furthermore, these recommendations can be found in any type of purchase or service, including what book you should read, what song should you listen or what movie should you watch.

According to a report presented by Grand View Research [5], the recommendation systems market had a value of around 1.7 billion dollars in 2020, and it is forecast to expand up to a 33% annually until 2028. The COVID-19 pandemic has lead many businesses to modify their way of operating their sales, but also has tailored many buying habits of customers into more online based purchases, leading to a need of thriving RSs. To put some light into the importance of Recommendation Systems, Adobe Research in 2021 showed that approximately a 34% of customers will shop more when receiving valuable recommendations based on previous acquires.[5]

In this section, several frequently applied methods for recommendations will be introduced and discussed. These approaches will be divided in content-based methods, collaborative filtering-based methods and hybrid methods. It is important to notice that all these recommendation approaches are applicable to multiple different tasks, and might not be the most suitable for the use case that is being tackled in this thesis. The chosen approaches will be explained in the following section.

2.1 Content-based Recommendation Systems

This kind of recommendation system is mainly based on the historical preferences of the users. These algorithms try to detect the most significant attributes of the users favorite items and are stored in the users profile. These attributes (content) of an already purchased or liked item by users are then matched to similar content items in order to generate a recommendation to the user [6].

One of the main advantages of Content-based Recommendation Systems is that they capture some specific interest of users, which can lead to recommendations of rare and unique items that might be of little interest to the majority of users. This approach can also be useful because they do not require much ratings or historical data, as well as co-occurrences of the items and users, also known as scarcity [7]. On the other hand, these approaches are based on known interests of users, so it can lead to bad recommendations for new users or might be limited in the ability to expand to new interests.[8]

This kind of approach has been used in a wide variety of applications of topics. Some examples of them are:

- Book recommendations with an embedding learned from Wikipedia content [9].
- Job recommendations from the M Recommender Systems Challenge Workshop 2017. The problem with new users known as *Cold Start* was tackled in this work [10].
- Group recommender systems, which are recommendations of items that are consumed socially by groups of people, instead of the more typical individual recommendations.

2.2 Collaborative filtering Recommendation Systems

The first recommendation systems techniques that were developed were called collaborative filtering (CF), introduced on 1992 [11]. These systems worked under the premise of making comparisons between the active user and other past user with similar product qualifications, assuming users who agreed in the past will see eye to eye in the future. These types of algorithms build systems from the existing data (e.g. ratings of products, user clicks, time on a web-page) which find similar preferences from past customers to the active ones.

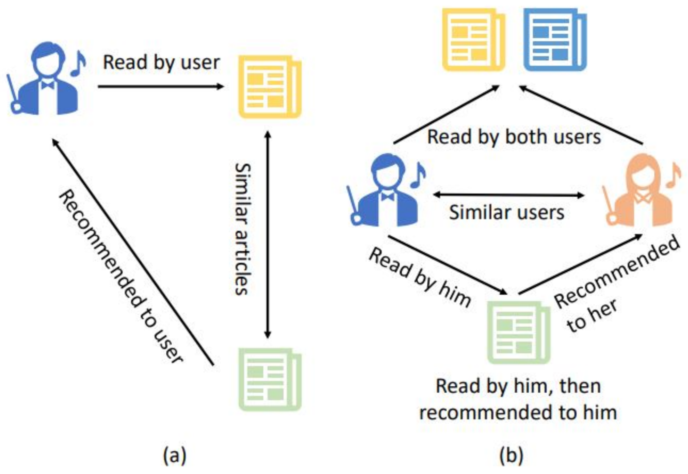


Figure 3: (a) Content-based Recommendation System (b) Collaborative Filtering-based Recommendation System.[8]

Collaborative filtering techniques can be split into two groups, depending on whether they are user-based or item-based. In the first case, the evaluation of the interest of users to an item is measured with the interest of similar users for that item. In the latter case, item-based approaches use the customer's past ratings of items to find similar ones as recommendations. Another classification on CF algorithms can be seen determined by the type of usage given to the data, depending on whether the data is used directly to make predictions (heuristic-based) or if the data is used to obtain knowledge to develop a predictive model (model-based).

2.2.1 Heuristic-based collaborative filtering

The first type of approaches are known as Heuristic-based CF, or neighborhood approaches, where the past interaction between users and items are stored in memory and are directly used for recommendations. These neighborhood approaches are defined as simple to implement, also leading to being comprehensible methods.

These methods don't require significant training routines, and are easily adaptable to new data or the addition of new items or users, both characteristics are helpful for many recommendation applications. However, the performance with this approach decays when the data is sparse, making these models not scalable.

Some of the algorithms that use heuristic-based collaborative filtering are:

- KNN (k-nearest neighbours) algorithm for news articles recommendations. [12]
- Prediction of rating of different items using Graph-theory approaches. [13]
- Internet shopping mall recommendation system with a Decision tree algorithm. [14]
- Movie recommendation for the EachMovie dataset with a Super Vector Machine (SVM), convined with a Genetic Algorithm (GA). [15]

2.2.2 Model-based collaborative filtering

These approaches are based on the use of some Machine Learning (ML) architectures in order to improve the results of the recommendations. The success of these approaches has tremendously risen over recent years and many collaborative filtering have adopted ML techniques for the enhancement of results. The idea behind these models is to capture the user-item hidden information and exploit it for better recommendations. This method can deal better with big amounts of data and usually outperform the heuristic-based approaches.

Nonetheless, these models might be hard to start from scratch, since they require large amounts of data, especially for ratings, which are usually hard to get in representative quantities. Subject to the data amounts or the desired model type, these approaches might require a lot of computational power, leading to costly solutions for recommendations.

In this section, some of the most popular ML algorithms applied as RSs will be introduced, including some work where they have been used:

- **Multi-Layer Perceptron (MLP)**: these networks are fully-connected feed-forward networks, with a number of hidden layers between the input and the output. These networks are the basics to many other more complex models, but they can also be used in many recommendation problems with successful results. [16]

- **Autoencoders**: AE are unsupervised networks that try to replicate the input at the output. These networks, usually convolutional or fully connected, have a bottleneck layer in the middle that can be used as a feature representation of the data. These architectures can be successfully applied to recommendation systems. [17]
- **Convolutional Neural Networks (CNNs)**: These networks are feed-forward networks connected by convolutions, allowing to capture location and neighborhood information. CNNs are designed for image treatment, but can also be used in plenty of areas including recommendations. [18]
- **Transformers**: These architectures are state of art models for natural language processing (NLP) problems, combining different types of attention, positional encoding and a encoder-decoder structure for excellent results, even in recommendation systems. [19]

2.3 Knowledge-based Recommendation Systems

For different problems on recommendations with scare user ratings or purchase history, like cars or apartments purchase, another type of RSs can be developed. In this case, these algorithms try to give users recommendation based directly on knowledge of users, items or relationships between them. The lack of need of a big dataset or purchase history makes this approach suitable for complex domain problems. However, the necessity of a domain knowledge required for the recommendation can be a limitation for different use cases. [8]

2.4 Hybrid-based Recommendation Systems

Some researchers have opted to combine both types of approaches in order to provide better recommendations by trying to overcome the drawbacks of each approach with the integration of them. For example, in 1999 an approach was presented that linearly combined the results of both approaches for better recommendations [20]. Another approach focused on integrating collaborative features (navigation or rating data) with content features under the maximum entropy principle for recommendations [21].

3 Methodology

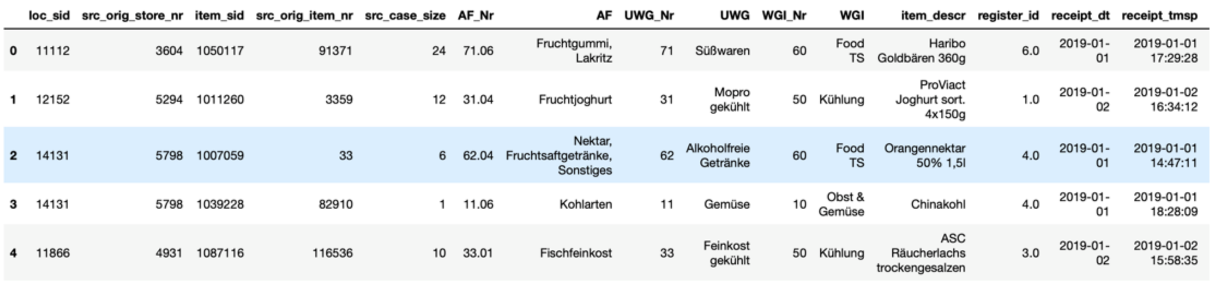
In this section two main components will be described. Initially, the dataset will be introduced, including a brief analysis of the available data. The second part will illustrate the different applied methods for the recommendations, including a definition of the implementation and the main advantages and disadvantages of each.

In order to perform all the computing, data analysis and model training among others, *Lidl* facilitated two virtual machines (VMs) from the STACKIT portal, an IT organitzation within the *Scwarz Group*, the largest retail company that includes *Lidl* and *Kaufland* [22]. These VMs are of 16 CPUs, with a 128GB RAM, without GPU access. This will be an important factor getting into some of the approaches, and will be touched also in the conclusions section.

3.1 Dataset

As mentioned in the previous sections, many recommendation systems are based on customer ratings on products or on features based on opinions of users (comments, clicks on web page or time spent on a web page). In this thesis the recommendations will be obtained from historical data of purchases, not ratings.

The used dataset is going to be a sizeable history of single items purchased in 15 different *Lidl* stores across Germany from January 2019 until August 2021. The following figure shows the format of some of the columns of this dataset.



	loc_sid	src_orig_store_nr	item_sid	src_orig_item_nr	src_case_size	AF_Nr	AF	UWG_Nr	UWG	WGI_Nr	WGI	item_descr	register_id	receipt_dt	receipt_tmsp
0	11112	3604	1050117	91371	24	71.06	Fruchtgummi, Lakritz	71	Süßwaren	60	Food TS	Haribo Goldbären 360g	6.0	2019-01-01	2019-01-01 17:29:28
1	12152	5294	1011260	3359	12	31.04	Fruchtjoghurt	31	Mopro gekühlt	50	Kühlung	ProViact Joghurt sort. 4x150g	1.0	2019-01-02	2019-01-02 16:34:12
2	14131	5798	1007059	33	6	62.04	Nektar, Fruchtsaftgetränke, Sonstiges	62	Alkoholfreie Getränke	60	Food TS	Orangennektar 50% 1,5l	4.0	2019-01-01	2019-01-01 14:47:11
3	14131	5798	1039228	82910	1	11.06	Kohlarten	11	Gemüse	10	Obst & Gemüse	Chinakohl	4.0	2019-01-01	2019-01-01 18:28:09
4	11866	4931	1087116	116536	10	33.01	Fischfeinkost	33	Feinkost gekühlt	50	Kühlung	ASC Räucherlachs trockengesalzen	3.0	2019-01-02	2019-01-02 15:58:35

Figure 4: Format for the *Lidl* Store Ticket dataset.

As it can be seen, the dataset is in German, a factor that slowed down some of the processes of the thesis as will be explained in the results section where a human subjective evaluation was involved. There are some columns that can be of great interest. The AF (article family), UWG (under merchandise group) and WGI (main *Lidl* group) columns refer to categories of product, ordered from more specific to more generic. The column *item_descr* is the one used as identifier of every different item, containing a total of 8128 individual products. However, for many of the approaches a reduced dataset was used, using only those products who appear at least 400 times in the dataset. That reduced the number of different products in half, which reduces the sparsity of the data. This reduced version of the dataset will be used in the approaches which require a training of a model, that will highly benefit from reducing the number of classes by half.

The dataset is composed of a total of 68.7 millions of purchases of individual products. Grouping the dataset by the store where they were purchased, the timestamp of the purchase and the supermarket line identifier that stored the transaction, the items that were purchased together can be obtained. With that operation, around 13M different shopping receipts are obtained, resulting into a mean length of slightly above 5 items per purchase.

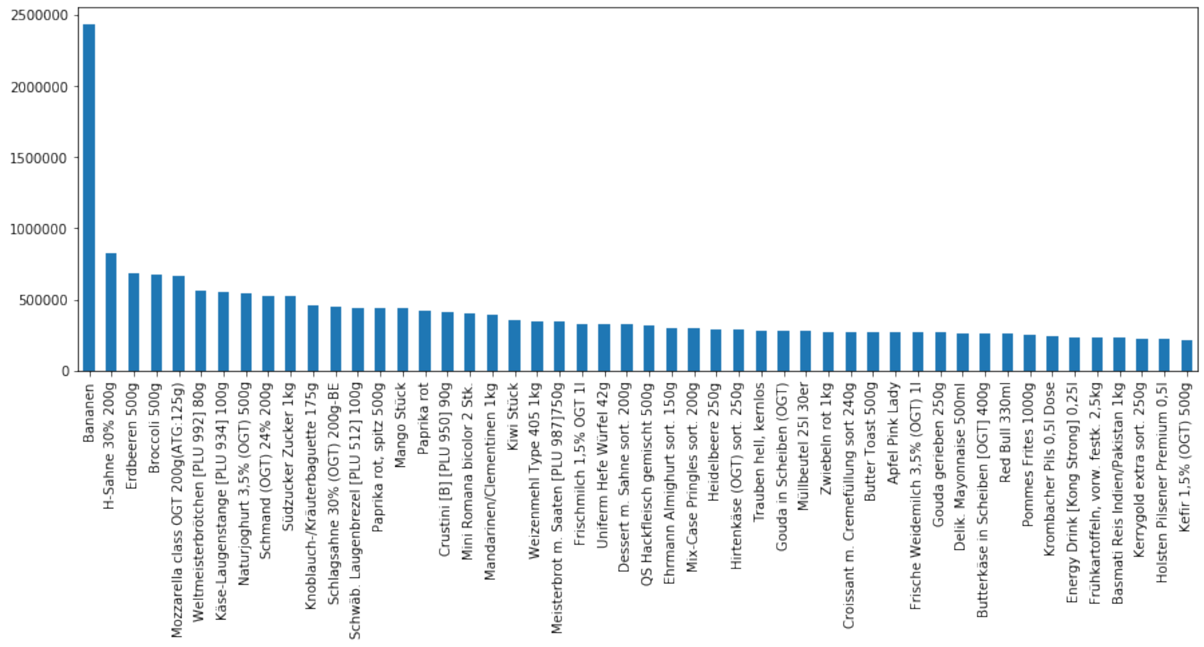


Figure 5: Histogram of top 50 purchased items from *Lidl* tickets dataset.

The previous histogram shows the top 50 most purchased items in the dataset. As it can be seen, there is a big unbalance between the top purchased product, starting with bananas (*Bananen*) that appear over 2M times, compared to it's next most purchased item, cooking cream (*H-Sahne*) with 0.8M appearances. The occurrences of all the products are disparate as already seen, since for instance only the top 15 purchased products cover up to a 10% of all the buys.

The following figure shows the histogram of the length of the receipts. As it can be seen, the majority of the purchases are smaller than 5 items, which makes them very suitable for the developed recommendation systems. However, a small number of the purchases have a large number of products, with a maximum of 188 different products in a unique purchase.

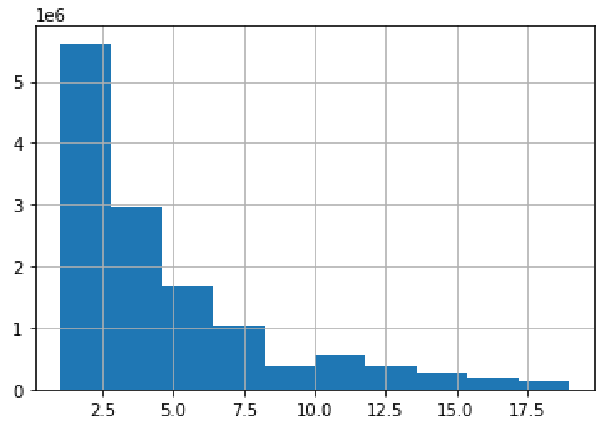


Figure 6: Histogram of the length of receipts.

As previously mentioned, this dataset is not developed to perform recommendations. Items that can be found together in a shopping list are not necessarily good recommendations for each other, which is a problem that will be explained more in depth in the results section. The dataset is also very large, which might lead to some memory problems for some of the possible approaches. In this case, some reduced or chunked versions of the dataset will be used.

One of the handicaps of the dataset is that all these receipts are from on-site purchases, not online. Therefore, there is no track of the sequential information of the purchases, that might be important for some of the approaches. Another obstacle found is that due to data protection issues, *Lidl* was unable to provide user information on shopping tickets. Even with a few customers registered, the dataset could have shown information of shopping habits of different buyers to create profiles of users. However, with the lack of this data, all the receipts were treated as from different individuals.

3.2 Applied approaches

In this section, different chosen approaches will be explained one by one, including some extra approaches that were not successful and were discarded. For each approach some of the obstacles and advantages will be explained without commenting on the results, that will be analysed in the next section.

3.2.1 Histogram-based approach

This first approach was based on histograms, which are graphical representations of groups of data in between some specified ranges. For this case, the histograms are developed from groups of items found together. For instance, the following figure shows the histogram of items that has been found with the items *Mango Stück* (mango unit) and *Spachtelset* (spatula set).

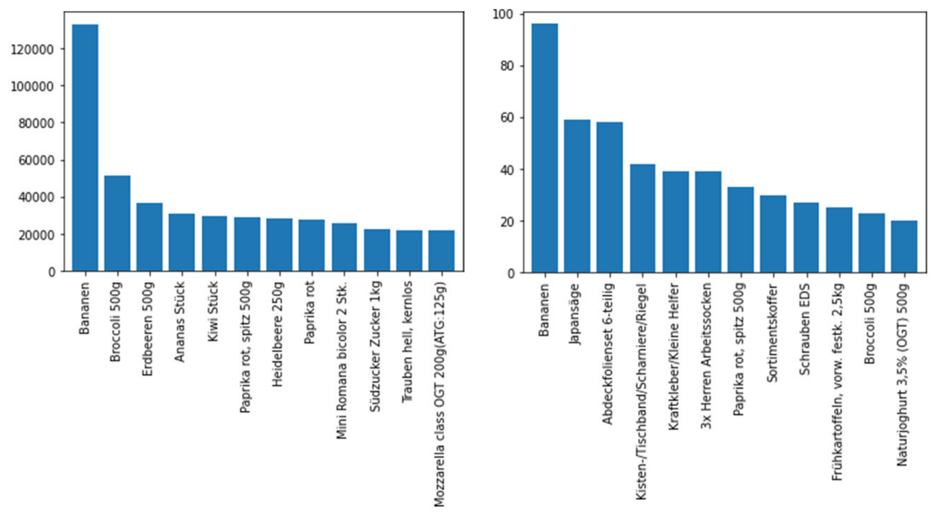


Figure 7: Histogram of top purchased items with *Mango Stück* (mango unit), on the left, and *Spachtelset* (spatula set),on the right.

It is interesting to see that, apart from bananas being a constant throughout the dataset, each of the items is commonly found with items of similar categories, for example the spatula set is found with: Japanese saw, a set of foil covers, hinges, glue, work socks and red paprika, which most of them are of similar categories or carpentry related items. On the other hand, the mango is found with broccoli, strawberries, pineapple, kiwis and paprika, all fruits or vegetables. As it can be seen, sometimes the most commonly purchased together items are from similar categories and would be valid for recommendations to each other. Nonetheless, there is a noticeable deficiency with the apparition of red paprika with the rest of carpentry items.

This approach was used as baseline to get introduced to recommendations. The main intention was to have a starting point from where to build the already mentioned chat bot, and get real examples obtained from the dataset, not just random selections. This technique was developed only for pair-wised items, so every time a user decided to purchase an item, a suggestion would be given to complement it instead of trying to supplement their complete shopping cart.

The reason why this approach is called histogram-based is that it is based on the number of occurrences of pairs of items. The probability of finding together two items will be estimated with the co-occurrences of them, that can be represented with histograms.

The way this approach proceeds is by computing a rating for every possible pair of items. For each existing product pair, we will compute an score obtained from two different terms. Let's suppose we have a product A, and we want to find the score of it being paired to product B. The first term will be computed with the times A and B have been found in the same purchase. This value will be divided by the times A have been purchased, as a normalization term. This term represents the probability of finding the product A if we have B.

The second term has the same numerator but it is instead normalized by the times B has been purchased, so it represents the probability of finding the product B if we have A. The reason behind this second term is that very common B items will get higher scores than less frequent items, due to the fact that it is normalized by the occurrences of A. With the addition of this second term, items B that are not found many times, but when they are purchased they can be found typically with A, will get higher scores for the recommendations. The sum between this two terms is weighted by a parameter α. This parameter can be tuned in order to get more common items as recommendations (higher α) or to benefit less typically found items. The algorithm can be defined with the following equation:

$$\text{Score} = \alpha \frac{\text{occurrences of A and B}}{\text{occurrences of A}} + (1-\alpha)\frac{\text{occurrences of A and B}}{\text{occurrences of B}} \tag{1}$$

In order to perform a recommendation for this method, when the user adds to the cart an item, the top rated items for the product selected by the user are chosen. The recommended item will be a weighted random choice from this top items, where the weights are obtained from the explained score. This randomness allows users to get different recommendations for the same items, in order to both test the recommendations and give some variability to the results. This approach allows users to get recommendations of pairs of items: for every purchased item, a recommendation for that item can be given.

This approach can be extended in order to cover any number of items in the shopping cart, not only pair-wised recommendations. The decided way to expand this approach is by using the mean of the score regardless the number of items selected. For instance, if we want a recommendation (item C) for items A and B, the mean between the C and A, and the C and B scores will be computed. The item C that has a higher mean score with all the other items in the cart will be selected as the better recommendation.

The main advantage of this approach is that these scores are not very computationally expensive to obtain. Thanks to the addition of the second term of the score, not very common items or products which are recently added to the dataset are easy to incorporate and are not a problem for this approach. The histograms can be easily updated regularly without the need of many resources, which makes this method very adaptable.

On the other hand, due to the inborn pair-wised naturality of this approach, it does not work very well for group recommendations. This factor will be better commented in the results section, but using the mean of the scores does not compensate along all the products. The recommendations observed with this approach are usually logical for one of the items in the cart, but not for the whole.

3.2.2 Graph-based approach

Graphs are mathematical representation of data designed to represent relationships between objects. It is believed that graph theory was initially introduced in 1736 by the solution of the Seven Bridges of Königsberg by Leonhard Euler [23].

These mathematical structures are composed by a set of nodes (also known as vertices or points) that characterize the objects that want to be represented. The relationship between two nodes is represented by a set of edges (also called links or lines). These edges represent the strength or the existence of the connection between the nodes.

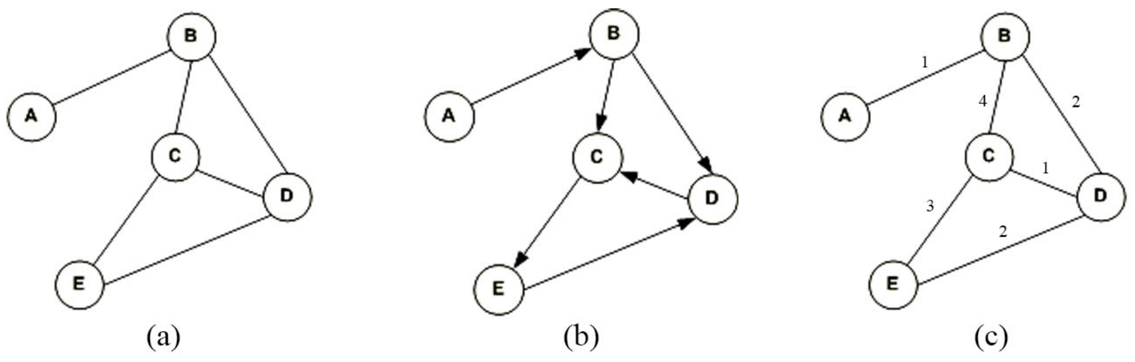


Figure 8: (a) Undirected graph (b) Directed graph (c) Undirected weighted graph.

Depending on the type of connections between the nodes, graphs can be split into directed graphs (or digraphs) or undirected graphs. In the first type, edges can have either one or two directions, which means that the fact that node A is connected to node B does not explicitly mean that node B will be connected to A. In opposition, edges in undirected graph do not require a direction, making the relation between nodes bidirectional. Edges can also have a weight, representing the strength of connections between nodes. These graphs are known as weighted graphs. The previous figure shows the difference between these types of graphs.

Graphs have been used for many years in a wide variety of implementations. Many publications have covered some of the state-of-art graph approaches for different applications, some remarkable are:

- Traffic forecasting: introduction to some graph approaches for traffic forecasting that have outperformed different Machine Learning approaches, including convolution neural networks and recurrent neural networks. [24]

- Brain networks: this paper reviews some graph-based popular neuroscience techniques, with salience to detection of network communities or modules, and the identification of central network elements. [25]
- River topology: Many successful simulation of rivers topology are performed with graphs. [26] reviews some of this simulations.
- Social networks: A very popular application for graphs is social networks analysis (SNA). This representations can show information circulation, business or interests networks and influence on users among many others. [27]

In this case, graphs are going to be used to perform recommendations. Many different approaches could have been made, depending on the type of structure desired, with different definitions of nodes or types of edges. In this section some different algorithms purposed will be presented depending on the type of connections between nodes, however, only the most successful one will be explained.

The used approaches took every different item as a node. The data protection issue caused to have over 13M different users, which made the option of having a node for every user unfeasible. The difference between the algorithms came on how the edges between the different items were set up. On the initial phases, the decision taken was to create an undirected and unweighted graph based on the mentioned score of the histogram-based approach. In this case, every node would have a fixed number of edges with the top nodes depending on the score.

The way these graphs were created was through the Adjacency Matrix, which is a square matrix that represents the connections of a graph. For example, the edge between nodes i and j is represented with the a_{ij} position of the matrix. The previous figure with 3 different graph would have the Adjacency Matrix that can be seen in the following equation. As it can be seen, only weighted graphs have non-binary Adjacency Matrix, and for the undirected graph, the matrix is symmetric.

$$\underline{\underline{A}} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 \end{bmatrix} \quad \underline{\underline{A}} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix} \quad \underline{\underline{A}} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 4 & 2 & 0 \\ 0 & 4 & 0 & 1 & 1 \\ 0 & 2 & 1 & 0 & 2 \\ 0 & 0 & 1 & 2 & 0 \end{bmatrix}$$

Figure 9: from left to right: undirected graph, directed graph and undirected weighted graph adjacency matrices.

In that initial case, recommendations would be computed from the number of steps that takes to go from the nodes of items that are being purchased to the recommended ones. The number of steps to get from any node to another can be computed with the adjacency matrix. There are a_{ij} possible paths of length k from i to j with the power of the adjacency matrix $\underline{\underline{A}}^k$. The intuition was to recommend the node that takes the least steps from all the purchased item nodes. For instance, if we had a shopping cart with 3 items, the number of minimum steps from every other possible node to this 3 items would be computed, getting a mean of the three lengths of the paths. The node that had the smallest mean distance would be selected as a recommendation.

Still and all, some mentioned premises where not desirable for the recommendations that have been corrected with the final approach. One of the modifications was to not take a fix number of edges for every node. The reason behind it is that it is totally acceptable that some of the items are never recommended, specially with the type of dataset available.

Some of the articles are purchased a few too many times to provide relevant information to get from them. Furthermore, with that initial approach, the graph ended up centered on only *Bananen*, which, as mentioned, is by far the most purchased item. The big majority of paths were through this item, and with that approach big part of the recommendations where *Bananen*. Another limitation of this approach is that the developed graph was not fully connected, making some of the nodes inaccessible. One solution to this limitation would be to establish a penalization for unconnected nodes, granting this distances a bigger value than any possible path between nodes in the graph. This solution did not fix the centralization problem of the graph, and the final graph's connections were computed differently.

This final graph was decided to be an undirected and unweighted graph as well. However, the connection between nodes are only obtained from the probability of finding both together, instead of the score, using a threshold instead of a fix number of edges per node. Since the connections are already bidirectional, it is not that useful to add the second term of the score equation. After some parameter tuning, it was decided to create an edge between node A and B if both of them appeared on the dataset at least 200 times (6000 different products where selected) and at least 0.05% of the times A is purchased it has to be with B.

The reason behind this parameters is the number of links created with them. After comparing some results, a reasonable number of edges was around a mean of 10 per node. With an slight decrease of this threshold to only 0.04%, the mean of nodes decreases from 9 edges per node to more than 15. In the other direction, if the percentage threshold was increased to 0.1%, the mean number of edges would fall to less than 3 per node. Therefore, a percentage of 0.05% of occurrences was selected as a the threshold for the creation of edges between nodes.

With the graph created, it was decided how the product suggestions would be performed. The way the recommendations will be evaluated will be explained more in depth in the next section, but in general terms, one of the focuses on how to consider a good recommendation is to be able to complement a couple of products in the cart.

Therefore, the way recommendations have been obtained in this approach is by looking for triangles of connected nodes in the graph. This triangles or 3-cliques have been of high importance in many applications of graphs, used to detect strong communities and its cohesiveness. They are also used to determine the graphs stability, and also for the computation of network indices like clustering coefficients. [28]

When two items that are part of a triangle are found in a shopping cart, the third node of that triangle will be the selected recommendation. In the case we have more than two products and multiple triangles can be obtained from the active shopping cart, if there is a more repeated node across all triangles, it will be selected, and if there is not, a random selection will be made.

As it will be seen in the results, this approach gives promising results and creates communities of items that are captured in the graph. Since the creation of the graph and the triangles is not very computationally expensive, the graph can be easily updated with new items or information obtained.

On the other hand, this approach requires at least having two already desired items before any recommendation can be given. It also has the limitation that all the selected items might not create a triangle with any nodes, which would also lead to no available recommendations. Since only having a mean of 9 edges per node might lead to few possible recommendations, a second auxiliary graph was created, using a lower threshold that leads to more connections. In case no triangle was found in the first graph, this second one would be used to provide an alternative recommendation.

3.2.3 Embeddings approach

The purpose of a word embedding is to represent words as vectors that can capture word semantics and syntactics so it can be used by a computer. It can be seen as a map of words into vectors, with the ability to capture interesting information. A trivial way (that it is not even considered an embedding) to create a vector representation for a word is to perform a what is known as one-hot encoding of words: for every known word, a vector of the length of every existing word is created full of zeros, with a 1 on the active encoded word. This approach gives no information on relationship between words, and neural networks can help on the creation of a significant embedding. By training a network with existing text, we can get significant mapping of the words into vectors.

The way the training of these networks is performed consists of trying to predict the next word of a sentence from the previous or the surrounding words (context). Therefore, the entry of the encoder is the already mentioned one-hot encoding of the context word, and the output of the decoder (and target of the prediction) is the one-hot encoding of the desired word. This word embedding can be trained in many other different ways, for example setting up the same input as output for the target for the network. With this approach, the network is able to learn a representation of the data in a reduced dimensionality (also known as latent space).

In the following figure an example of this kind of structure is shown. The vectors that represent the words are the values that the encoder outputs in the middle layer. The dimension of this vectors can be tuned depending on the desired solution, and determines the size of the embedded vector.

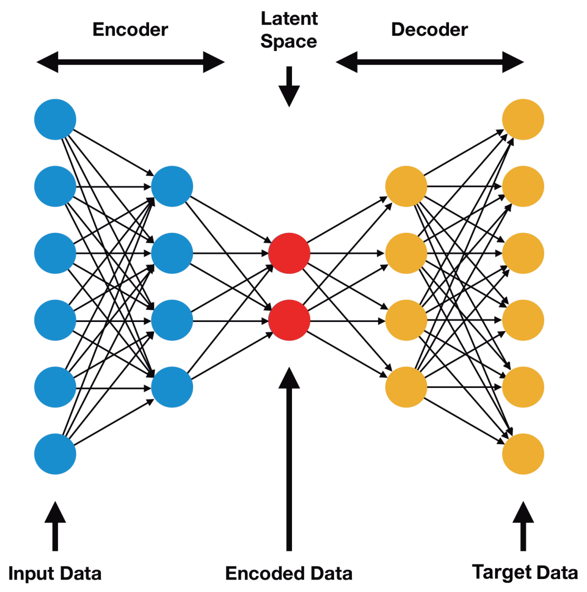


Figure 10: Example of an Encoder-Decoder system [29].

These vectors are able to capture many semantics and syntactics of the words. One of the most important features is that words that have similar semantics have smaller Euclidean distance representations that words that do not. The corresponding points of similar semantic words are expected to fall into closer points in the latent space than the representation of semantically different words.

These semantics are captured in the positioning of the words representations, for example, for successful embeddings, the vector difference between words like *man* and *woman* is very similar to the difference vector between *king* and *queen*, both in distance and direction. In a similar way, syntax can also be captured, and can be shown with very small distance vectors between word pairs that share syntactic meaning, like the subtraction of the vector representation of the words *walking-walked* should be very similar to the subtraction of *swimming-swam*.

This Encoder-Decoder system can be adapted to this recommendation use case. In this case, the entry is not going to be a one-hot encoding of a word but a representation of a receipt. As vocabulary, all the existing products could be used. For this approach, only items that appear 400 times were used, which resulted in about 4000 different products. The training consisted of trying to replicate the input to the network at the output, with a bottleneck of neurons in the central layer in order to get the latent space representation. For every receipt, a one-hot encoded training vector was created, having the same input to the network as expected output (label). The length of the input and output vectors is the number of different products find in the store, in this case around 4000, filled with ones for the present items and zeros for the missing ones for every receipt.

The training of this approach was also tested in a more classification oriented way, where only one of the items in the receipt is the target of the prediction. After trying this approach, it was discovered that the target vector was too sparse, since it contained 4000 zeros and only a single one. This was attempted to be fixed with the usage of other available information on the dataset, like the families of the products.

A developed approach was to instead of giving 0 to the items that were not the target, give a small reward if at least the category was correct. The AF (subcategory), UWG (category) and WGI (family) where incorporated, and for instance, if the target was *Heidelbeeren* (blueberries), some score at the target would be given to all the products in the subcategory *Beeren* (berries), some to the ones in the category *Obst* (fruit) and some to the ones belonging to the family *Obst & Gemüse* (fruit and vegetable). This score will be inversely proportional to the size of each of the groups.

This second approach with the families information was unsuccessful due to the lack of memory of the used machine. The dataset without this information can be stored in boolean variables (true or false) since it is binary data. However, the information of families requires float types of variables, that are typically used with 32 bits (can provide 7 decimal digits). This big addition to the memory requirement, meant a very impactful reduction of the dataset, which ended up causing worse results than the simple binary approach.

The selected approach consisted of an autoencoder structure, trained trying to replicate the input at the output with one-hot encoding of the receipts. Both the encoder and decoder had 2 hidden layers, of 2000 and 500 neurons, with a latent space of 12 dimensions, performing batch normalization after each of the layers. All the layer used a ReLU activation, except for a softmax for the last layer.

One of the main advantages of this embeddings method compared to the other approaches is that it is way more flexible with the input and output. In this case, we can get a recommendation for any number of ingredients, and it is not limited like the other methods. The return of the network is also a ranking of all the products as a recommendation to that entry, thanks to the activation function selected at the last layer, which allows flexibility in order to give recommendations, like recommending with a weighted average for the top items or simply selecting the top product.

With the bottleneck of the encoder-decoder we can also get a representation in a latent space in order to see how similar products are when they have smaller euclidean distances in order to find communities or groups of ingredients.

The way recommendations will be performed is by using the representation of the actual shopping cart in the latent space. The closest item, with euclidean distance, to that point will be the one selected as the best recommendation. A ranking of every single item can be obtained by computing the euclidean distance to the embedding representation of the cast.

Nonetheless, these models are hard and computationally expensive to train, which makes them not very suitable for the addition of new items to the approach. New ingredients would also require a redesign of the neural networks, and might lead to bad results if they are not retrained from scratch every time.

3.2.4 Support-vector machine approach

A support-vector machine (SVM) is a machine learning approach introduced in 1993 by Corinna Cortes and Vladimir Vapnik. The implementation idea is to train a model able to create a decision surface (known as support-vector) that splits into two regions the inputs of a classification problem. The data points are represented into a high dimension feature space, under the premise to split them into classification regions minimizing a loss function. [36]

At their origin, SVM were designed for binary classification, however it can be also used in multiclass classification. The main solution to generalize to multiclass classification is to transform the approach into multiple binary classifications, usually with a one-versus-one method [37]. This approach consists of training a binary model between every possible pair of classes. The final prediction is the most voted class among all the binary problems.

As it has been seen, SVM are models typically used on classification problems. In our case, we can transform the recommendations into classifications as seen in the embedding-based approach. The training of the model is going to be performed by predicting one random item of every receipt (target) with the other items on the receipt as input.

Even though this approach has been used in other recommendation systems [15], the results were not successful for this use case. These models are not the most efficient for classification with large numbers of classes, which lead to poor results. One of the main issues found was that the big majority of the trainings resulted into the model mostly classifying every entry to the top class, *bananas*. The one-versus-one approach favored the most common class, making it the final prediction for a the big majority of the test examples. This outcome didn't provide bad metric results, however it was not a desired solution.

Nonetheless, the approach will be considered in the results in order to show the big importance of the class *Bananen* in this dataset. The model learns that recommending almost always that item, gives a good objective result to the recommendation. In the results section, the performance of recommending *Bananas* predominantly will be analyzed

3.2.5 Other attempted approaches

In this section two attempted approaches will be presented. Due to different barriers that will be introduced, these models won't be presented in the results, but can achieve state of art results for recommendations in other use cases.

3.2.5.1 Transformers

In 2017 a Google team proposed a new and revolutionary architecture called transformer [30] , that are based on attention and convolutional and artificial neural networks. This architecture is a Sequence-to-Sequence (or Seq2Seq) model, which means that transforms a sequence of words into another sequence. This architectures achieved state of art results in plenty of NLP Seq2Seq tasks, such as machine translation, document summarizing or text generation. [30]

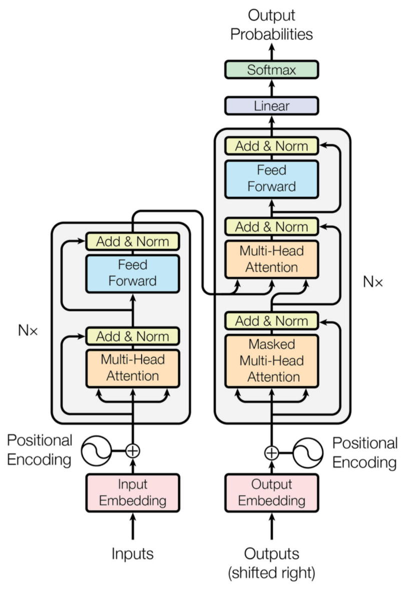


Figure 11: Transformers structure.[30]

As it can be seen in the figure, transformers are also based in a encoder-decoder system. As an addition, this models have a positional encoding, which is the information of the relative position of each word in the sequence. This is required since there is no recurrent neural networks that can remember the order in which the sequence is introduced to the model. This information is added to the embedding representation of each word.

These transformers have stepped up as the most successful models for NLP with the appearance of some pretrained systems, like BERT (Bidirectional Encoder Representations from Transformers) or GPT (Generative Pre-trained Transformer)[31]. Thanks to the parallelization of transformers, these two models were pretrained on large datasets and can be used for plenty of tasks with just a small fine-tuning.

As it was already mentioned, these models are trained with a positional encoding, since they are very good at dealing with large sequences of words. However, the used dataset has the limitation of not knowing the sequence in which the items are purchased. Instead all the products are found together in the same shopping list. Transformers can be really good for recommendations because they are used in for example book reading or films recommendation. Clearly in this cases there is some temporal dependence of the users, that can not be exploited in this use case.

Another obstacle found when using transformers is the Multi-Head attention layers, where the words have to be embedded into a significant mask. The models usually use pretrained embeddings or tokenizers in order to use the dataset for the actual approach. The names of the products of the dataset could be used as the word to tokenize. However, these names were not valuable since a lot of information was about the units or content of the products (*Orangennektar 50% 1,5l*), or where brand names which are not part of many tokenizers.

For these two main reasons transformers were not successfully trained for this use case and were discarded as a viable approach for the *Lidl* recommendation system.

3.2.6 Decision tree approach

Another tested classification method for this problem was decision trees, which are predictive models used in a wide variety of applications, including for example medicine [32], economics [33] or computer vision [34] among many others.

These models work under a set of successive conditions based on the database in order to classify samples into the different classes depending on several input variables. The models begin with an initial node (root), where one of the input variables is compared to the root attribute. Depending on the result of the comparison, one of the branches is followed to the next node, repeating this process until a leave is reached. Each leaf is associated to a class, which is chosen for the sample when the leaf is reached.

One of the main advantages of decision trees as classification models is its simplicity and interpretability [35]. The results provided by a decision tree are easy to follow and understand, simply by following the decision taken at each of the nodes. On the other hand, the training of this trees can be unstable, understood as the fact that a small change on the training data can result in to a major change to the decision tree structure.

Similarly to the SVM based approach, this classification method can be used as recommendations for this project. The model has been trained with the same approach, where one item is predicted from the other items on the receipt.

The results obtained from this classifier are also very similar to the ones obtained by the SVM approach. These models are unable to predict the missing product of the receipt, and fall into the triviality of always recommending *Bananen*. Even this results is the one achieved for a better loss function for this approach, it is not a desirable solution, and since SVM have a very similar performance it was a discarded approach.

4 Results

In this section the results of the recommendation systems will be presented and discussed. Before jumping into the analysis of the results, the way of evaluating the recommendations will be explained.

4.1 Evaluation of recommendations

One of the main question marks around the presented problem is: *What is a good recommendation?* For this use case, there is no direct answer to that question, which has been a strong barrier for this thesis. Usually, datasets used for the creation of recommendation systems contain direct opinions on the products, like ratings on books, users watching or not a recommended film among many others. This information is usually related to some users that consume the services or products periodically.

In this case, the data was directly history of purchases, which has no direct connection to what is a good or bad recommendation or any user data. The data was also very unbalanced to one main item, bananas. According to the data, this is a good item to recommend in a lot of purchases, since it can be found in many different shopping carts that seem to not have any correlation with bananas. For example, there are multiple cases of purchases of non-food items (e.g. cleaning soaps) with bananas, however, from a recommendation point of view, it is not very reasonable.

In order to provide results, two tests will be performed. The first one is going to be an objective evaluation for the different approaches. A number of receipts will be chosen, randomly removing one of the items and selecting it as target. The algorithms will provide a recommendation with the other part of the receipt, and only if it matches the removed item it will be a successful recommendation.

A second test is going to be performed to the approaches, where subjective judging is going to be made on the recommendations in order to give them validity or not. The recommendations will be classified into three different categories: good, regular or bad, humanly judging if it exists a correlation or a motive to recommend the item purely based on the original items.

In order to facilitate the tagging of the recommendations, it was decided to take always two different items as input to get one recommendation. Only using pair-wise recommendations was too simple and would not help demonstrating good knowledge of the approaches. Nonetheless, if there is a clear relationship between the two items and the item recommended by the approaches it can be considered as a successful recommendation. If more than two items were used, the recommendations might be too vague and would highly increase the difficulty of tagging the results.

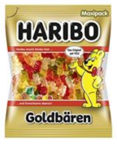
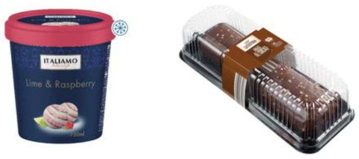
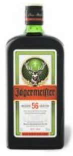
Evaluation	Original item	Recommended item
Good recommendation		
Bad recommendation		
Regular recommendation		

Figure 12: Recommendations rating example.

The previous figure shows some examples of how the recommendations are going to be rated. If there is a clear correlation between the items and what is recommended, like in the case of tea bags and milk with sugar, the recommendation will be accepted. If the two items have nothing to do with the recommended item, like recommending candy to someone buying a surf board and broccoli, the recommendation will considered as bad.

A better recommendation for that case could be for example an energetic drink or some healthy products. In some cases the relation will be fair but not good, like recommending an alcoholic drink for ice cream and cake. Some other dessert products might be a better recommendation, even though the alcohol might be a great recommendation if the user is planing a party or social event. These cases will be tagged as regular recommendations.

Since this is a very costly and time consuming way of evaluating the results, the test sets will not be very large. For every approach around 100 recommendations will be evaluated, as it is considered a significant enough size for comparing the results and performances of the algorithms.

As a baseline to compare to, the original dataset was passed through the same test all the approaches will do. A random selection of about 100 tickets of only 3 items where picked up, setting two items as original items and the third one as recommendation for the others. It was seen that only 44% of the recommendations were considered as good, with a 20% as regular and a 36% as bad. As it can be seen, the dataset that should contain the *ground-truth* for the recommendations, fails to provide good results. Therefore, this problem can be considered more as a noise cleaning of the dataset more than a standard classification or recommendation problem.

4.2 Approach performances

The performances will be divided into the two previously introduced sections. The first one is going to be an objective evaluation of the recommendations, followed by a subjective analysis of the performance of the results.

4.2.1 Objective evaluation

In order to objectively test the results, a total of 200 thousand tickets were randomly selected. For each of them, one random item has been selected as target, and the algorithm are tested to predict it with the other items on the receipt. The selected size of the test is representative enough to show the performance of the approaches and small enough to be performed with the mentioned time and memory limitations.

The receipts were a random selection, without any preprocessing, only with the limitation of requiring a minimum length of two items, one as target and one as input. This resulted into some approaches not being able to provide recommendations for some of them. For instance, some examples only had items that are very few times, which were eliminated in some of the approaches. For the receipts with only 2 items, the graph approach was not able to provide recommendations, since no triangles could be found.

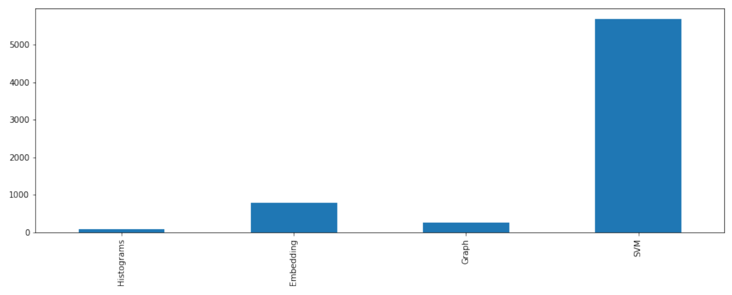


Figure 13: Objective results for the applied approaches.

As it can be seen, the SVM approach outperforms all the other approaches, getting the right recommendation in almost 3% of the tested receipts. However, this percentage is very similar to the 3.5% of times that *Bananen* appears on the dataset compared to the total of receipts (2.4M times out of 68.7M products).

The embeddings is the next better performing approach with an accuracy of 0.4%. It is important to notice that the embedding works with around 4000 classes, and a random selection of a target would provide around 0.2% accuracy. The other two approaches perform even worse than a random selection of a product.

As it can be seen, the objective evaluation is not a reliable method for the analysis of the performance of this use case. Only choosing Bananen on all the test samples provides a good metric score, but it is not a well founded recommendation system. The other approaches have very poor performances for this evaluation and do not provide valuable information.

4.2.2 Subjective evaluation

In this section the subjective performance of the different approaches will be compared. All of the techniques will get a similar test already mentioned, where a recommendation will be asked for two different items about a 100 times. The main goal of all approaches is to be able to outperform the baseline dataset performance.

The following image shows the performance of the mentioned test of all the developed approaches compared to the results for the original dataset.

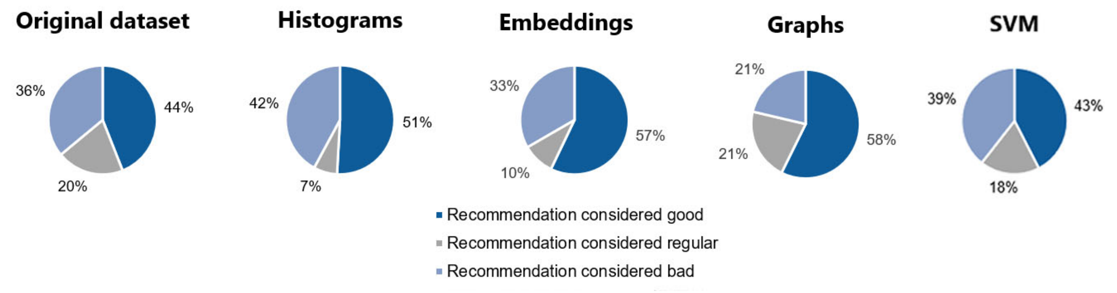


Figure 14: Results for the applied approaches.

As it can be seen, all the proposed approaches are able to clean the noise from the dataset by outperforming the original dataset results. The approach with better performance in this test is the graph approach, with a remarkable only 21% of bad recommendations, followed by the embeddings and finally by the histograms.

It has to be said, that the graph approach obtains the better results, but at the same time, is the only one that can not provide results for every different shopping cart. For around a third of the tested pairs for the original dataset results, the graph approach was not able to provide a recommendation, due to the lack of existence of triangles.

A simple combination of two approaches could be made in order to get the more realistic results for the graph approach, using for example embeddings when there is no recommendations for graphs. This would slightly drop the accuracy of graphs, but would still leave it as the better performing approach.

The only method that performs very similarly to the original dataset is the SVM approach. This algorithm recommends the majority of the time the item *Bananen*, which by being the most dominant (by far) item on the dataset might be a good recommendation for lots of receipts. However, using this subjective evaluation, this recommendation is not considered valuable in almost 60% of the times, failing to provide good recommendations.

One factor that is not completely visible in the results is the quality of the recommendations. Even they are classified into 3 groups, the suggestions of products can be outstanding sometimes, or on the contrary, a complete non sense. These two types of recommendations are not visible from these percentages but are present in the results. Using more divisions than the basic 3 mentioned would make the tagging of the results too challenging and even more subjective, so leaving the classification into good, regular or bad was considered the most optimal.

During the evaluation of the results, some of the outstanding recommendations were marked when classifying, being the graph approach the one that got more of them by far. For instance, this approach was able to capture branding items together, and when two items from the same brand were purchased, a third one would typically appear as a recommendation (e.g. *Bebivita* baby products or *Barilla* pasta products). For about the 30% of good recommendations for that approach, it was marked as surprisingly good, and it only happened sporadically in other approaches. Nevertheless, this is due to the naturality of the approach, tested by taking existing triangles for the evaluation.

This precision of finding 3 very commonly found items together was only correctly captured by the graph approach. Some individual test examples were taken, in order to compare the results of the approaches for the same receipts. These, were always of length 3, and with the input of 2 to the approaches, a recommendation was going to be compared to the original third item of the receipt.

An example is a receipt that contained sugar and bananas. The predictions of each approach were a chocolate bar (histograms), red onion (embeddings), strawberries (graph) and bananas (SVM). This last one was not considered a possible recommendation, as the item was already part of the receipt. The real third item of the receipt was a pack of cigarettes, which clearly would be considered as a bad recommendation for the other two items. In this case, the graph approach outperforms the other methods with a very good recommendation, having a noticeable relationship with the two products. The chocolate bar was considered a regular recommendation, as it was a sports bar, more healthy-related than sugar. Finally, the onion was categorized as a bad recommendation, due to the lack of consistency with the other items.

Another factor to comment is that the training of embeddings can be improved with hyperparameter tuning. Due to the lack of big computing power, the training of the models took very long times (around 10 to 15 hours depending on the parameters), which lead to a lack of time to deeply explore the networks. The other approaches have a close ceiling, since histograms and graphs can be improved but won't modify much the performance. Instead, the training of the embedding can be upgraded by finding a more optimal skeleton of the autoencoder or by having different trainings routines. Therefore, it is considered that with deeper study of the embeddings, this approach might be able to outperform the other methods.

These results are way more significant than the ones found in the objective evaluation. One of the main inconvenience of the dataset was the lack of a better objective and fast way to judge the performance of the approaches. Evaluating the recommendation was a time consuming task, and might be biased by a lack of large testing sets, which does not guarantee that the graphs are really the best general approach for recommendations with this dataset.

5 Conclusions and future development:

To summarize, different approaches for recommendations have been built from a dataset that clearly was not designed for that task. With different approaches, the valuable information buried in the history of shopping tickets, has been successfully obtained, ending up with an approach with around 60% considered good recommendations.

The proposed graph approach is able to outperform in the explained test all the other approaches, reaching reasonable recommendations for the existing dataset. This approach is able to capture relationship between products covered in the dataset that enhance the recommendations. With a simple algorithm development, this approach is adaptable to recommendations for large shopping carts, with also flexibility for the integration of new items. The creation and development of the graph is not very computationally expensive, which makes it a great solution to the use case.

The usage of a CPU (instead of GPU) based computer and the lack of high memory or computing power, limited the performance of some of the approaches. With a better infrastructure, the training of the models could have been faster and better exploit, and might have lead to, at least, improved results.

The model based approaches highly benefits over the others with big amounts of data availability. Some convenient information like seasonal information, or even some offers are simple to incorporate to these approaches. However, due to the lack of computing power, models had to be kept simple and were surpassed by an heuristic approach.

The main barrier that was found in this thesis was the lack of an objective, fast and valuable way of evaluating the problem. The proposed objective evaluation did not show the real performance of the recommendation systems of the different approaches. Even though the subjective evaluation was an effective solution, results might vary if the evaluation was performed by another individual, and the size of the test might not have been extensive enough to guarantee which of all the approaches had the best performance.

One of the future developments proposed is to set up a production environment, where real users can be presented with the proposed recommendations. If *Lidl* applies this system integrated in a chat bot, users can interact with the recommendations and determine if they like or not the suggestions by purchasing or discarding the items. After some time of collecting this data, a new model could be trained with the interactions of users with the suggestions, which would lead to a dataset where users decide to purchase or not a recommended item.

Other possible upgrades to the project could be to test new approaches that were not successful in this thesis, like the transformers or the decision tree based approach. As it has been seen, all the proposed approaches are based on collaborative filtering, however, the names of each of the items could be processed with some natural language techniques in order to create a hybrid approach for better results.

As a conclusion, four different recommendations systems have been built from a supermarket receipts dataset, with implementation details, advantages and disadvantages of each of them. The best performing approach is the graph-based approach, with very promising results on recommendations. The approach is very flexible and is chosen as the most viable technique for the *Lidl* supermarket use case.

Acknowledgements

I would first like to thank Prof. Brian Subirana for giving me the opportunity to do this thesis abroad, where I found one of the best experiences of my life. I would like to extend this gratitude to Mirko Saul and Valentin Häußer, who opened me the doors to the development of this project. I wanted to thank everyone in the MIT Auto-ID Laboratory for making my stay so comfortable and make me feel like home, specially to Niklas Thum, who helped me through the struggles of the thesis. Furthermore, I wanted to give a special thank to Prof. Sanjay Sarma who stepped in the project to solve the doubts when it did not have a clear path and also made my stay at the lab much easier and enjoyable. Finally, I want to express my very profound gratitude to the two supervisors of this thesis, Dr. Stephen Ho and Dr. Montse Pardàs, for being my big help and guidance during the whole development of the project. Without their advice, support and willingness to help I wouldn't have been able to complete the thesis.

References

[1] Dr. Shahid Bhat, Keshav Kansana, and Jenifur Majid. A review paper on e-commerce. 02 2016.

[2] Stephanie Chevalier. Retail e-commerce sales worldwide from 2014 to 2025. 02 2022.

[3] Alon Esisenberg. Discovery trust: Uk online shopping survey. 09 2019.

[4] Reuters Staff. German discounter lidl slows u.s. expansion: paper. 2 2018.

[5] Grand View Research. Global recommendation engine market size, share trends analysis report by type (collaborative filtering, hybrid recommendation), by deployment, by application, by organization, by end-use, by region, and segment forecasts, 2021-2028. page 100, 09 2021.

[6] Pasquale Lops, Marco de Gemmis, and Giovanni Semeraro. *Content-based Recommender Systems: State of the Art and Trends*, pages 73–105. Springer US, Boston, MA, 2011.

[7] Yilena Pérez-Almaguer, Raciel Yera, Ahmad A. Alzahrani, and Luis Martínez. Content-based group recommender systems: A general taxonomy and further improvements. *Expert Systems with Applications*, 184:115444, 2021.

[8] Yuanzhe Peng. A survey on modern recommendation system based on big data, 05 2022. Online; accessed July 2022.

[9] Cataldo Musto, Giovanni Semeraro, Marco de Gemmis, and Pasquale Lops. Learning word embeddings from wikipedia for content-based recommender systems. volume 9626, pages 729–734, 03 2016.

[10] Maksims Volkovs, Guang Wei Yu, and Tomi Poutanen. Content-based neighbor models for cold start in recommender systems. In *Proceedings of the Recommender Systems Challenge 2017*, RecSys Challenge '17, New York, NY, USA, 2017. Association for Computing Machinery.

[11] David Goldberg, David Nichols, Brian M. Oki, and Douglas Terry. Using collaborative filtering to weave an information tapestry. *Communications of the ACM*, 35(12):61–70, dec 1992.

[12] Paul Resnick, Neophytos Iacovou, Mitesh Suchak, Peter Bergstrom, and John Riedl. Grouplens: An open architecture for collaborative filtering of netnews. In *Proceedings of the 1994 ACM Conference on Computer Supported Cooperative Work*, CSCW '94, page 175–186, New York, NY, USA, 1994. Association for Computing Machinery.

[13] Kun-Lung Wu Philip S. Yu Charu C. Aggarwal, Joel L. Wolf. Horton hatches an egg: A new graph-theoretic approach to collaborative filtering. New York, NY, USA, 1999. IBM T. J. Watson Research Center.

[14] Yoon Ho Cho, Jae Kyeong Kim, and Soung Hie Kim. A personalized recommender system based on web usage mining and decision tree induction. *Expert Systems with Applications*, 23(3):329–342, 2002.

[15] Sung-Hwan Min and Ingoo Han. Recommender systems using support vector machines. volume 3579, pages 387–393, 12 2005.

[16] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. Neural collaborative filtering. In *Proceedings of the 26th international conference on world wide web*, pages 173–182. arXiv, 2017.

[17] Hao Wang, Naiyan Wang, and Dit-Yan Yeung. Collaborative deep learning for recommender systems. In *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1235–1244. arXiv, 2014.

[18] Suhang Wang, Yilin Wang, Jiliang Tang, Kai Shu, Suhas Ranganath, and Huan Liu. What your images reveal: Exploiting visual contents for point-of-interest recommendation. In *Proceedings of the 26th International Conference on World Wide Web*, WWW '17, page 391–400, Republic and Canton of Geneva, CHE, 2017. International World Wide Web Conferences Steering Committee.

[19] Qiwei Chen, Huan Zhao, Wei Li, Pipei Huang, and Wenwu Ou. Behavior sequence transformer for e-commerce recommendation in alibaba. In *Proceedings of the 1st International Workshop on Deep Learning Practice for High-Dimensional Sparse Data*, pages 1–4. arXiv, 2019.

[20] Anuja Gokhale and Mark Claypool. Correlation thresholds for more accurate collaborative filtering. In *Proceedings of the IASTED International Conference Artificial Intelligence and Soft Computing*. Citeseer, 1999.

[21] Xin Jin, Yanzan Zhou, and Bamshad Mobasher. A maximum entropy web recommendation system: combining collaborative and content features. In *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, pages 612–617, 2005.

[22] Schwarz IT KG. We are stackit!

[23] E.; Wilson R. Biggs, N.; Lloyd. *Graph Theory, 1736-1936*. Oxford University Press, Oxford, UK.

[24] Weiwei Jiang and Jiayun Luo. Graph neural network for traffic forecasting: A survey. *Expert Systems with Applications*, page 117921, 2022.

[25] Olaf Sporns. Graph theory methods: applications in brain networks. *Dialogues in Clinical Neuroscience*, 20(2):111–121, 2018. PMID: 30250388.

[26] Finnbar Lee, Kevin S. Simon, and George L. W. Perry. River networks: An analysis of simulating algorithms and graph metrics used to quantify topology. *Methods in Ecology and Evolution*, n/a(n/a).

[27] Alessia D'Andrea. *An Overview of Methods for Virtual Social Network Analysis*, page 8. Springer, 2009.

[28] Luca Becchetti, Paolo Boldi, Carlos Castillo, and Aristides Gionis. Efficient semi-streaming algorithms for local triangle counting in massive graphs. In *Proceedings of*

the 14th ACM SIGKDD international conference on Knowledge discovery and data mining, pages 16–24, 2008.

[29] Steven Flores. Variational autoencoders are beautiful. *Comp Three Inc.*, Apr 2019.

[30] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2017.

[31] Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, and Amr Ahmed. Big bird: Transformers for longer sequences. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 17283–17297. Curran Associates, Inc., 2020.

[32] Kasra Madadipouya. A new decision tree method for data mining in medicine. *Advanced Computational Intelligence: An International Journal (ACII)*, 2(3):31–37, 2015.

[33] Marek Durica, Jaroslav Frnda, and Lucia Svabova. Decision tree based model of business failure prediction for polish companies. *Oeconomia Copernicana*, 10(3):453–469, 2019.

[34] Raphaël Marée, Pierre Geurts, Justus Piater, and Louis Wehenkel. A generic approach for image classification based on decision tree ensembles and local subwindows. In *6th Asian Conference on Computer Vision*. Asian Federation of Computer Vision Societies (AFCV), 2004.

[35] Anthony J. Myles, Robert N. Feudale, Yang Liu, Nathaniel A. Woody, and Steven D. Brown. An introduction to decision tree modeling. *Journal of Chemometrics*, 18(6):275–285, 2004.

[36] C. Cortes and V. Vapnik. Support vector networks. *Machine Learning*, 20:273–297, 1995.

[37] Kai-Bo Duan and S. Sathiya Keerthi. Which is the best multiclass svm method? an empirical study. In Nikunj C. Oza, Robi Polikar, Josef Kittler, and Fabio Roli, editors, *Multiple Classifier Systems*, pages 278–285, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

Appendices

In this section. the developed chatbot will be introduced. This chatbot was a baseline of work in order to get an idea of what a real *Lidl* customer would have when getting recommendations. This algorithm will get further developed as a Master Thesis of another partner at the lab, and only the initial state, developed by the author of the thesis, will be introduced.

The dialog flow was developed with the *Cognigy AI* software, a company that provides techniques to automate customer and employee communications. This tool allows users to create rule-based conversation flows with box-diagrams as seen in the following figure.

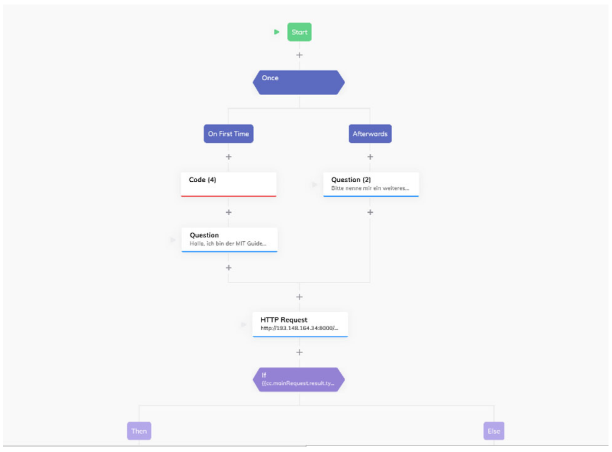
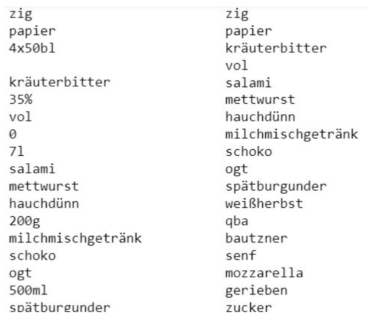

Start
Once
On First Time
Afterwards
Code (4)
Question (2)
Bitte nenne mir ein weiteres...
Question
Hallo, ich bin der MIT Guide...
HTTP Request
http://193.148.164.34:8000/...
If
{{cc.mainRequest.result.ty...
Then
Else

Figure 15: Start of the chatbot flow diagram.

Initially, the user is asked to introduce the name of a product to add to the shopping cart. The problem is that it is hard for users to exactly know the name of an existing product of the *Lidl* database. In order to solve it, a list of words was developed. This list consists of every possible known word found in all the possible names of products in the dataset.

As it can be seen in the following image, the list of known words was very noisy and not practical. As mentioned before, a lot of the names of the product did not provide reliable information on what the customer was looking for, like including the weight or the volume of the different products. In order to clean it, some regular expressions were applied, trying to preserve the valuable information and deleting all the unnecessary words.

For each of these known words, all the items that contained that word in their name were added to candidates when that word was typed by a user. The post-processed list of words was also amplified with the information of the families, categories and subcategories available from the dataset. For example, for the word *Beere* (berry), all the items belonging to that category were added to that word.



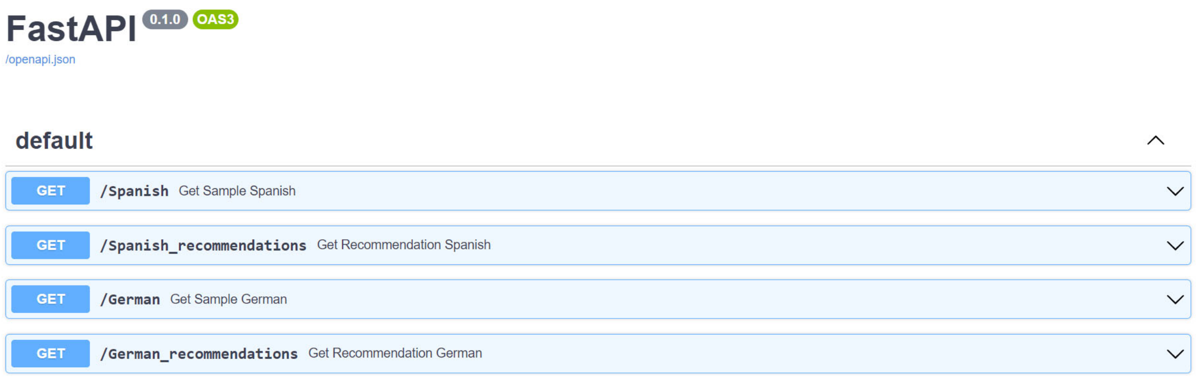
```
zig                       zig
papier                    papier
4x50bl                    kräuterbitter
                          vol
kräuterbitter             salami
35%                       mettwurst
vol                       hauchdünn
0                         milchmischgetränk
7l                        schoko
salami                    ogt
mettwurst                 spätburgunder
hauchdünn                 weißherbst
200g                      qba
milchmischgetränk         bautzner
schoko                    senf
ogt                       mozzarella
500ml                     gerieben
spätburgunder             zucker
```

Figure 16: Initial list or words (left) and post-processed list (right).

When the user was asked to type what item they were looking for, every word in the input was searched in the post-processed list of known words. The candidate that had most words in common with the input is going to be the one added to the cart. In case there is a tie between different products, the user will get a list of the possible items in order to choose one of them, suggesting always the most commonly purchased first.

Every time a new item is added to the cart, the user will be given a recommendation of an extra product to be added as well, using the most successful recommendation method, in this case the graph approach. If the graph can not provide a recommendation for any of the reasons previously presented, the user will get a recommendation only dependent on the last addition to the cart with the histogram approach, which has proven to be a successful recommendation system for pair-wised items.

As mentioned, the models are running on a VM at the STACKIT portal. On this machine, there is a docker image that is able to build a FastAPI. The structure of this API can be seen in the following image.



FastAPI 0.1.0 OAS3
/openapi.json

default

GET `/Spanish` Get Sample Spanish

GET `/Spanish_recommendations` Get Recommendation Spanish

GET `/German` Get Sample German

GET `/German_recommendations` Get Recommendation German

Figure 17: FastAPI structure.

This API is able to respond to the mentioned petitions from the chatbot. As it can be seen, there is both a Spanish and a German version. This Spanish version was an initial test, since an initial, very reduced, Spanish version of the dataset was given in order to start developing the chatbot.

The way the *Cognigy AI* chatbot is able to communicate with the API is by HTTP requests. Depending on the type of petition (recommendation or list of products) performed by the user, the server is able to respond with a dictionary with the desired information.