FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Location Aware Product Recommendations

Pedro José Leal de Sousa

Mestrado Integrado em Engenharia Informática e Computação

Supervisor: Rui Camacho (FEUP)

Co-Supervisor: Luis Moreira (Fraunhofer Portugal AICOS)

July 22, 2017

Location Aware Product Recommendations

Pedro José Leal de Sousa

Mestrado Integrado em Engenharia Informática e Computação

July 22, 2017

Abstract

Nowadays, a typical retail chain store catalog encompasses thousands of products, the sheer quantity of products makes it difficult for the customer to be familiar with all the options and their specificities without spending too much time in each shopping trip. In order to make products known that the customer may be interested, while providing potential store sales, recommendation systems are applied to reduce the information examined by the customer and help him decide alternatives, to explore other products and categories that may please him. With the vast customer knowledge that stores already have, it is possible to extract information such as preferential products, shopping patterns, product related categories and understand what can make a better shopping experience for the customer. Recommendation systems can be applied to any store type, usually traditional recommendation systems based on collaborative or content-based filtering use simple models. Context-aware recommenders take into account not only the customer purchase history but the context in which those purchases were made, and also takes into account the target user current context when generating recommendations. One possible context is the user's location and whereabouts inside the store, with this type of information it is possible and desirable to use it to produce better, more personalized and timely (well-timed) product recommendations. The final product of a recommendation system should be considered as a powerfull personalized assistant who knows the customers and all the products of the store, and during their shopping trips, advises and guides them according to their tastes and, in this case, their location. Taking advantage of Fraunhofer AICOS previous experience and know-how in the areas of accurate internal location and product recommendation, these two techniques were combined into an innovative solution that helps improve customers planning and shopping trips offering counselling before and during the customer journey. Context-aware recommendation systems was explored combined with periodic and sequential pattern mining in order to build a robust shopping companion app and support system.

Resumo

Hoje em dia, um típico catálogo de lojas de retalho engloba milhares de produtos e essa vasta quantidade de produtos dificulta ao utilizador a perceção de todas as opções e suas respetivas especificações sem gastar muito tempo em cada viagem de compras. Para dar a conhecer potenciais produtos ao cliente e simultaneamente favorecer potenciais vendas em loja, os sistemas de recomendação são aplicações que reduzem a informação analisada pelo cliente e ajudam a decidir alternativas, ao explorar outros produtos e categorias que possam ser do seu interesse. Com o vasto conhecimento sobre o cliente que as lojas já possuem, é possível extrair informações como preferências do utilizador, padrões de compras, categorias relacionadas com produtos previamente comprados e, portanto, entender o que pode melhorar a experiência de compra para o cliente. Os sistemas de recomendação podem ser aplicados a qualquer tipo de loja, geralmente sistemas de recomendação tradicionais baseados em filtragem colaborativa ou baseada em conteúdo usam modelos simples. Os context-aware recommenders tem em conta, não só o histórico de compras do cliente, mas também o contexto em que essas compras foram realizadas e o contexto atual do utilizador alvo ao gerar recomendações. Um contexto possível é a localização do cliente e a posição dos artigos dentro da loja, com esse tipo de informação é possível apresentar melhores recomendações, mais personalizadas e oportunas. O produto final de um sistema de recomendação deve ser considerado um poderoso assistente personalizado que conhece os clientes e todos os produtos da loja sendo capaz de os aconselhar e orientar de acordo com seus gostos e, neste caso, a sua localização durante as suas viagens de compras. Aproveitando a experiência e o know-how da Fraunhofer AICOS nas áreas de localização interna precisa e recomendação de produtos, essas duas técnicas foram combinadas numa solução inovadora que ajuda a melhorar o planeamento das viagens de compras dos clientes oferecendo aconselhamento antes e durante o seu percurso. Foram explorados sistemas de recomendação com conhecimento de contexto combinados com a extração de padrões periódicos para construir uma aplicação de acompanhamento de compras robusto e um sistema de suporte.

Acknowledgements

This master thesis work could not be possible to do it alone.

First, I would like to thank FEUP for during my academic journey to have encouraged me to seek more knowledge and in particular to professor Rui Camacho for all the guidance and help during this process.

I would also like to thank Fraunhofer Portugal for the opportunity to work on this topic and for all the support given, especially to my mentor Luis Moreira and to David Ribeiro for all the help and guidance.

I would also like to thank my colleagues, especially my companions Mário Ferreira and Alcino Sousa for the company on the train trips and for all the fun times that we spent together.

I would like to give a big thank you to my parents, my brothers and sister, especially to my brother Carlos Sousa for the spell check and his wife Ana Cavaleiro for keeping me focused on this thesis.

Last but not least, friends who I am sure will stay for life and who have always been present whenever I needed, Francisco Couto, André Regado, Rui Gonçalves and Antonio Presa.

Pedro Sousa

"Restlessness is discontent and discontent is the first necessity of progress.
Show me a thoroughly satisfied man and I will show you a failure."

Thomas Edison

Contents

List of Figures

LIST OF FIGURES

List of Tables

Abbreviations

AR	*Association Rules*
BuilT	*Build Time*
CARS	*Context-Aware Recommender System*
CAS	*Context Aware Systems*
CF	*Collaborative Filtering*
CSS	*Cascading Style Sheets*
CSV	*Comma-Separated Values*
HTML	*HyperText Markup Language*
JSON	*JavaScript Object Notation*
LARS	*Location Aware Recommender Systems*
MAE	*Mean Absolute Error*
MBCF	*Memory Based Collaborative Filtering*
MSE	*Mean Squared Error*
NMAE	*Normalized Mean Absolute Error*
NDCG	*Normalized Discounted Cumulative Gain*
RMSE	*Root Mean Squared Error*
RS	*Recommender System*
SVD	*Singular Value Decomposition*
SVG	*Scalable Vector Graphics*
TestT	*Test time*
TSP	*Travelling Salesman Problem*
WWW	*World Wide Web*
XML	*eXtensible Markup Language*

Chapter 1

Introduction

In this Chapter we present a review of the dissertation's context. The problem domain will be explored together with the motivations and the expected result. For an easy navigation through this report the structure is explained in the last Section.

1.1 Problem Domain

The fast evolution and development of networking infrastructures boosted e-commerce to new levels of importance. With this growth, a huge quantity of products/services and encompassing information became available to customers. To be able to confront this information overload recommendation systems are presented as one possible solution [PG13b].

Recommender systems allow the dynamic manipulation of data received, taking into consideration the target user preferences. The main goal of recommender systems is to personalize content in a scalable way, offering options or alternatives. These options depending on the business could mean a wide range of services such as products, friends, news. The research on recommender systems aims to show that personalized systems can improve the user experience avoiding the overload of information and present with confidence services that the user may be interested in.

Recommender systems are present in most e-commerce services, suggesting new experiences with potential and to assist the costumer [PG13b]. One of the most common user-experience that the recommender systems provides is the personalized page, "Recommended for you" where the website suggests a set with the *topN* relevant items to the client.

Stating an explanation/cause when providing recommendations to the user may influence the user's ratings. Studies demonstrated that explaining recommendations can create a trust connection with the user while other studies found out that users tend to be more satisfied with systems that seem to know what they like [CLA+03].

1.2 Motivation and Goals

Mobile applications and computing systems are attracting the interest of researchers and industry due to the potential huge impact on people's daily lives, and by the advances in artificial intelligence [HK15].

Nowadays, mobile devices are considered to be the primary choices of information access [PG13a]. In the market there is a wide range of applications with integrated recommender systems that capture attention from users due to their importance in helping on their daily decision making (example: what movie to watch, where to eat) or simply discovering new services that they may be interested in.

Due to the necessity of personalization, the use and evolution of mobile technologies, recommender systems can capture more knowledge about the user helping to find better solutions for the costumer.

With the intention to take advantage of one of the resources that mobile technology makes available, the location of the user, we aim to improve the recommendations generated. Even though most machine learning algorithms cannot be directly used on mobile devices due to processing power limitations that still exist today, the advantages and the opportunity of business that the location provides is already appreciated for example in tourism services [PG13a].

The final goal is to develop a recommender system that takes into consideration the user's current context, focusing specially on his spatial information.

1.3 Document Structure

This document is structured as follows. Chapter 2 reviews the state-of-the-art in recommendation systems as well as addresses some related work in the field. The problem being targeted and some strategies to solve it are presented in Chapter 3. Chapter 4 reviews the development of the tool for Location Aware Product Recommendation System and then in Chapter 5 it is present the tests made to this system and examples of results extracted. Chapter 6 presents the work plan and the conclusion of this master thesis and thoughts about future work on this matter.

Chapter 2

Recommender Systems

This Chapter introduces the background and related work on recommendation systems. It is important to begin with traditional recommender systems to be able to understand the origins of more complex systems. Throughout this Chapter we can retain how the recommendation systems became more complex and tried to take advantage of the technologies for a better adaptation to the current user needs.

Well designed recommender systems that consider item features and the user's preferences is able to recommend new items that are suitable for the user, improving his experience.

Traditional recommendation approaches provide, in an efficient and effective way, recommendations based on user's and item's informations. With the growth of information about the context and the necessity to produce more appropriate recommendations, new approaches have been proposed to deal with cases where traditional systems answer wasn't enough.

2.1 Traditional Recommender Systems

Traditional recommendation approaches (collaborative filtering and content-based filtering) excels in quality and taste recommendation tasks such as movies and books. Yet, in contextual situations such as financial services traditional approaches are not the best choice [RRSK11].

2.1.1 Collaborative Methods

Collaborative filtering recommender system (CF), provides recommendations to user based on the opinions of similar users. The main idea is to analyze the user behaviours to predict which items the user may be interested. In a pure CF scenario [SMU14], the only information considered in this approach, is the user-item matrix, which contains the ratings that the users gave to the items they experienced.

Collaborative methods can be divided into two approaches:

1. Model based methods

   Model-based approaches create a model based on collection of user ratings of items previously consumed. This approach uses machine-learning techniques, such as classification and clustering algorithms to learn the recommendation model. A great variety of models have been successful in this approach, for example:

   - Association Rules: Using association to build an Collaborative Models based on commonly occurring patterns in the user-item matrix [Lin00]. Association rules are represented by the form $\{X\} \Longrightarrow \{\mathrm{Y}\}$, where X and Y represent different itemsets. There is two measures that helps the selection of rules from the set of all possible rules: support and confidence. Support of a rule is the percentage of transactions that contain both X and Y through all past transactions. Confidence of a rule denotes the percentage of transactions containing X which contain also Y.

     Table 2.1: Transactions table [Lin00]

Transaction Id	Purchased Items
1	{A,C}
2	{A,C,B}
3	{A,D}
4	{B,E,F}

     Consider the Table 2.1 and the rule $\{A\} \Longrightarrow \{\mathrm{C}\}$. The support of this rule is 50% calculated by the following formula:

     $$support(\ \{A\} \Longrightarrow \{C\}\ ) = p(A \cup C)$$

     The Confidence of this rule is 66% calculated by the following formula:

     $$confidence(\ \{A\} \Longrightarrow \{C\}\ ) = \frac{support(\ \{A\} \Longrightarrow \{C\}\ )}{p(A)}$$

     The antecedent and consequent side of the rule represent itemsets, which could contain any number of items. This approach can also be adapted to only discover rules with certain items that the costumer are interest in [Lin00].

   - Matrix Factorization: usually used when the density of the dataset is very low (high sparsity), which means only a few items are rated. MF models are based on latent factor models gathering user and item characteristics through item classification history [KBV09].

     Each item i of the dataset is associated with a vector $q_i \in \mathbb{R}$ that measure the factors in a positive or negative way, and each user u is associated with a vector $p_u \in \mathbb{R}$ that represents the interest the user has in items that are high on the corresponding factors

[KBV09]. The vectorial product between q_i^T and p_u represents the user's overall interest in the item's features denoted by r_{ui}. The recommender system have now the ability to estimate the rating that the user will give by using the following formula:

$$r_{ui} = q_i^T \; p_u \approx r_{ui}$$

Recent research suggests modelling directly the ratings avoiding overfitting through a regularized model that reduces the difference between the true rating and the approximated rating for all products and users [KBV09].

$$\min_{q*,p*} \sum_{(u,i)\in K} (r_{ui} - q_i^T \, p_u)^2 + \lambda(\| \, q_i \, \|^2 + \| \, p_u \, \|^2)$$

- Clustering Methods: The main idea of clustering is to divide the data in groups (clusters) based on their similarity. There are multiple clustering methods in data mining but, one of the most important is K-means [UF98] adapted to the CF method showed in Figure 2.1.

Input: Dataset *User-Item Rating Matrix*, number of clusters k
Output: Set of cluster centers C, cluster membership vector S
- Initialize cluster centers C
- **Do{**
// Data Assignment
  - Find closet cluster center for each user in User-Item Dataset using Minkowski distance and update S such that s_i is cluster ID of $user_i$.

// Relocation of centers
  - Update C such that c_i is mean of users in ith cluster}

While (C doesn't change)

Figure 2.1: Algorithm K-means [DM11]

Using clustering methods to predict item ratings involves three steps [DM11]: First we do *User clustering*, diving the users based on their similarity giving as input the matrix user-item; Second, giving a user u, we search for the cluster that the user belongs to; The final step is to find the most frequent score for a item i in the cluster that the user u belongs.

2. Memory based methods

   Memory based collaborative filtering (MBCF) uses samples or the entire history of the user-item dataset to generate a prediction for an potential item. The primary task of MBCF

is to determine the groups and which users belong to each group, based on similar their preferences.

Memory based collaborative filtering can be divided into two approaches, both of them are portrayed in Figure 2.2.

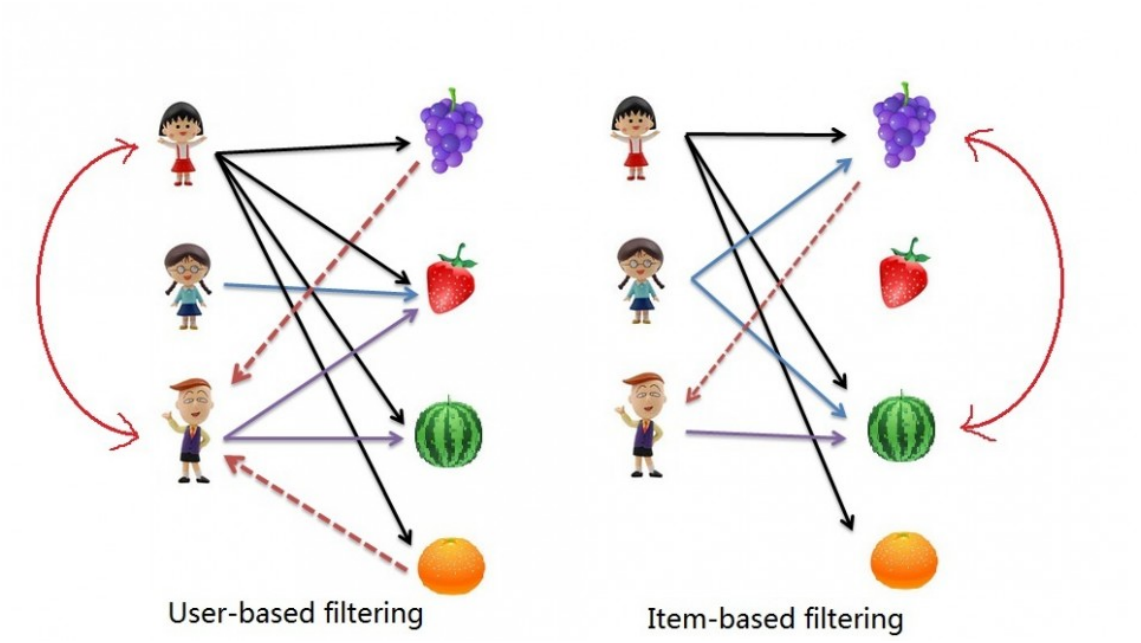


Figure 2.2: Collaborative Filtering Memory base[1]

(a) User-based Collaborative filtering

User-based recommendation algorithms generetate recommendations for users based on history of ratings of items given by users who present similar taste to the target users. The process of this method begins with a target user u_1 and a set of users $U = (u_2, u_3, ..., u_n)$, then it is generated predictions for an item i by analyzing ratings for i provided by the users U who are similiar to u_1. This approach suffers from three aspects [Ote15] : First, matching users with few common classifications is prone to distorted correlations. Secondly, it fails to recognize different relevance of agreements between users. Third, In the discovery phase of the user's neighbours on a large system, it is computationally expensive to compare with all users. This model is not best suited for large scale systems, but is more efficient in recommendations with a generous number of users.

(b) Item based Collaborative filtering

In item based approach unlike the user based filtering, the predicted items are generated by the ratings of similar items. There are two phases on this approach: First, is it

[1] http://www.salemmarafi.com/wp-content/uploads/2014/04/collaborativeFiltering-960x540.jpg [June, 2017]

calculated the similarity between all pair of items using similarity metrics. Second, is produced a list of recommendations, based on similar items to the already user liked. This approach does not scale well on large volume of items, even though Sarwar et al. [SKKR01] developed a solution, reducing the size of items by using topN-correlated items with *k co-ratings*.

2.1.1.1 Disadvantages of Collaborative-based Filtering

Collaborative-based filtering, even thought can produce accurate recommendations presents some limitations [Ote15]:

- New Item: CF to be able to recommend a certain item needs enough ratings for the target item to recommend accurately to a user. If the target item has not been rated cannot be considered for recommendation, potently losing some interesting and new choices to the users.

- Sparsity problem: Sparsity problem occurs when the number of ratings to predict is more than the history of ratings given to the model. This problem can be bypass using hybrid recommendation techniques or using models such as SVD to provide lower ranks of the original matrix.

- Intrusiveness: The data provided to this model had to be given to the system by the users, something that can be related to privacy problems, when users don't really know what information the system gain or why and how it works.

2.1.2 Content-based Methods

Content-based approaches analyzes the features and descriptions (if possible) of the items previously reviewed by a user, in order to build a user profile able to distinguish his interests. In this case there's no matching with other users neither between items, but rather exploit information about the item's features presents in the data[BV09]. The process of content-based systems involves three steps, handled by a separate component [BV09]:

1. Content Analyser: frequently is needed some pre-processing in order to be able to obtain relevant information from datasets. The purpose of this phase is to represent the item's features in structured data (e.g movies represented as keywords vectors), to be able to pass as input to the next phases.

2. Profile Learner: with the user preferences collected and through supervised machine learning techniques, it is created a generalization of this data in order to build a user-specific model. This user profile relates user interests (ratings) to item attributes, stored in a profile repository to be used in filtering component.

3. Filtering component: the purpose of this final phase is produced recommendations on items for specifics users, using the learned model from the previous step. This step must represent fast computing performance to be able to run in real time. The output is a ranked list of potentially relevant items for the user.

2.1.2.1 Advantages and Disadvantages of Content-based Filtering

Comparing Content-based recommendation against the collaborative approach we can identify some advantages [BV09]:

- User Independence: while collaborative filtering require ratings provided by other to calculate the *nearest neighbours* of the target user, content-based exploit only the ratings provided by the user in consideration to build his user model.
- Transparency: it is easy to understand how the recommender systems works, by identifying the content features or similar descriptions to the items that the active user liked. Collaborative on the other hand only gives the information that it was based on other users with similar tastes.
- New Item: Content-based approach can recommend new items without any previous rating (avoids ramp-up problems), something that the collaborative recommenders isn't able without the rating history. By relying only on user's preferences new items can be recommended without a substantial number of ratings.

Comparing Content-based recommendation against the collaborative approach we can identify some drawbacks [BV09]:

- Limited content: content-based techniques require enough information about the items. The representation of the items may only capture certain aspects of the content, but there are a lot of factors that can interfere the user's experience. The descriptions and features presents in the data may not be sufficient to highlight important aspects that users may be interested in.
- Over-specialization: The degree of novelty can be very limited, by only recommend items with high similarity between them. Sometimes the user may be interest in something new.
- New user: To be able to a profile user, the model requires enough ratings to capture user preferences and predict accurate recommendations (ramp-up problem). Therefore, without enough information about the user's preferences the system will struggle to predict good recommendations to new users.

2.2 Hybrid Recommender systems

Hybrid recommender systems combine at least two results from different recommendation techniques to work around the drawbacks of each recommendation technique. The most common way

to create a hybrid recommender is to combine collaborative filtering with some other technique to avoid the ramp-up problem [Bur07]. The use of hybrid strategies isn't a easy task in some situations, most datasets don't allow to compare different paradigms, lacking information such as item features, domain knowledge, requirements. There are some combination methods that stand out and will be mentioned in the following Subsections.

2.2.1 Weighted-Hybrid Recommenders

Weighted hybrid recommender scores a weight for each recommendation from the recommendation techniques presents in the system. The simplest case for this approach would be a linear combination from all the recommender's output. There are some improvements to this specific case where initially all recommenders have the same weight, but gradually adjusted (training) as predictions about users are confirmed [Bur07]. The main advantage of weighted hybrid is that can combine all recommendation system's capabilities on the recommendation process in a simple way and if necessary adjust the hybrid accordingly.

2.2.2 Switching-Hybrid Recommenders

In switching hybrid recommenders there is a switch between recommendation techniques when the current one is struggling for example in case of ramp-up problem. Typically is used content-based first, if there isn't sufficient confidence on the recommendations output, collaborative recommendations is attempted. The main concern about this recommendation process is to define the switching criteria which introduce complexity into the process. The advantage of this process is the sensibility that the system gain to the strengths and weaknesses of the recommenders that are used.

2.2.3 Mixed-Hybrid Recommenders

Mixed hybrid recommenders are used when there is a need to create large number of recommendations. The recommendations from two or more techniques as combined together, usually it removes the drawbacks from single techniques but not all, the ramp-up problem involving new users is common across multiple techniques.

2.2.4 Cascade-Hybrid Recommenders

Cascade hybrid recommenders involves integrates stage processes, where the recommendations are refined. First is produced a coarse ranking of recommendations candidates from one TR followed by a refinement from a second technique. This way the system avoid apply recommendations techniques on items that will never be recommended (poorly-rated). Compared to Weighted-Hybrid, Cascade-hybrid is more efficient because it discards low-priorities items, only refines items with potential.

2.2.5 Meta-Level-Hybrid Recommenders

Meta-Level-Hybrid uses a model (as a whole) generated by a RS as input for another technique. This model has the advantage in content/collaborative hybrid of processing user's interests following by a collaborative process that operates on this compacted data (avoiding sparsity problems) becoming more easy to represent than on raw data.

2.3 Context aware recommender systems

2.3.1 Context

Context presents multifaceted concept that has bring attention from multiple different research disciplines (eg. Computer science, philosophy, linguistics) [AT10]. A well-know business research named C. K. Prahalad has affirmed that "the ability to reach out and touch customers anywhere at anytime means that companies must deliver not just competitive products but also unique, real-time customer experiences shaped by customer context"[AT10], so we can see how the context matters so much to stand up against competitors. By being study from multiple areas, context has gain multiple definitions across various disciplines but the focus in this dissertation is on recommender systems, which is correlated with some areas [AT10]:

- Data mining: Contextual information in data mining is commonly used as information that can define events that characterize the stages of a costumer which can influence changes across his preferences and status. By knowing the context of the costumer, it is possible to mining patterns only for that specific context and selecting only relevant results that can be applicable to that specific context.

- E-commerce Personalization: Palmisano et al. defines context in E-commerce the intent of a purchase made by a customer. The main idea is that the costumer can use the same account for different purposes/contexts. To specify the intent of each purchasing motives, Palmisano et al. suggests the build of separate profile for each purchase [AT10] (identifying the context if possible), then use the separated profiles to build separate models having in consideration the context and specific segments of customer. In E-commerce, recommender systems has a significant impact in certain personalization degree, authors have demonstrated that considering the contextual information in the recommendation process along with the typical user-item matrix, can improve the quality of recommendation in certain sceneries.

- Mobile context-aware systems: The definition of context in mobile RS was originally defined as location of the target user, taking into consideration the users around and the objects around [AT10]. Although many other aspects have been added to this definition such as date, season, temperature. Contextual information has been crucial in multiple mobile-commerce, where the location for instance can be used to alert discounted tickets for a short duration of time.

- Marketing and Management: Prahalad [AT10] defines context as "the precise physical location of a customer at any given time, the exact minute he or she needs the service, and the kind of technological mobile device over which that experience will be received.", focusing on delivering unique, real-time customer experiences based on contextual information. Prahalad divides contextual information into three dimensions: temporal (when to deliver customer experiences, now or in the future), spatial (where to deliver customer experiences and technological (how to deliver customer experiences). Context information can bring a lot of new possibilities to improve user experience and consequent the revenue of business.

2.3.2 Representing and Modeling Context

Usually, the generation of recommendations has been seen as a predictive problem, predicting the user's rating for a specific item taking into consideration the user profile and items available:

$$\mathrm{R} : Users \times Items \longrightarrow Ratings$$

Once system has enough ratings provided by users or deduced by the system, it is estimated the function R, and consequent is predicted all the (User,item) which doesn't have an associated rating. Traditional system is viewed as two-dimensional (2D) since it only takes two dimensions (User and Items) into account in recommendation process. Context-aware recommenders systems (CARS) deal with recommendation problem by incorporating available contextual information, modeling and predicting user preferences and interests not only based on items and users but also with the context. Example: we can have information about movies and about users, but, if we had information about time the user has to spent on the cinema, or with who the costumer is different movies could be suggested.

$$\mathrm{R} : Users \times Items \times Context \longrightarrow Ratings$$

Context can represent multiple factors that delineate the circumstances under which the (User,Item) pair was assigned to a particular rating [AMRT11].

2.3.3 Obtaining Contextual Information

There are three main ways to obtain contextual information:

- Explicitly: Approaching directly relevant stakeholders interest in the recommendations by asking to fill a form or to answer some questions before any recommendation process (example: ask what kind of meal the user want).

- Implicitly: Implicit contextual information is extracted from the environment, such as the location of the user, or implicitly obtained from the transactions such as temporal context information.

- Inferring: In order to obtain this information data mining (example: Naive Bayes classifiers and Bayesian Networks) or statistical methods must be used. Example: identify if the user is walking or or in a vehicle by speed patterns.

2.3.4 Paradigms of Context Aware Systems

Traditional recommender systems use as input tuples in the form of (User, Item, Rating), in contrast CARS receive as input tuples in the form of (User, Item, Context, Rating) with contextual information about in which context the item was experienced (example: context = Monday).

The contextual information can be used in different phases of the recommendation process, leading to three different approaches of CARS [AMRT11] as it is possible to see in Figure 2.3.

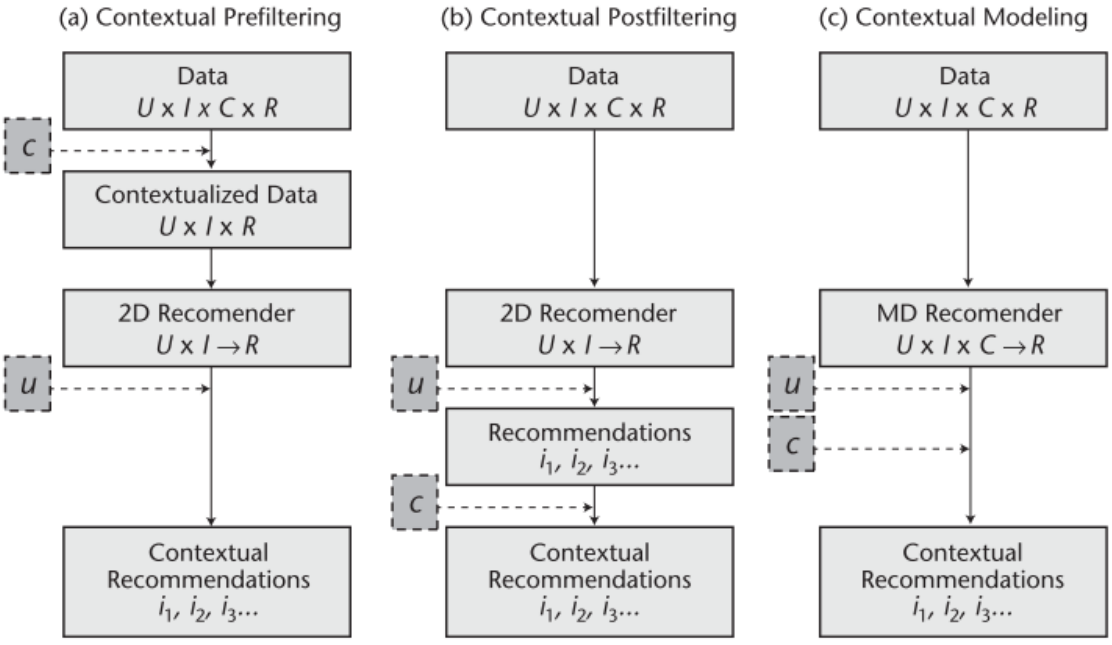


Figure 2.3: Paradigms of Context Aware Systems [AMRT11]

- Contextual Prefiltering: The contextual information C is only used as a filter to select relevant data, used after as input to any traditional 2D recommender system. One major advantage of this paradigm is that allows the use of any traditional recommendation techniques. The context information is used as query/filter for selecting only relevant rating data, for example: if the user wants to watch a movie on Monday, only movies that will be aired in Monday will be considered.

- Contextual Postfiltering: Unlike the previous paradigm, contextual information C is initially ignored. After the use of any tradition 2D RS the output is adjusted (contextualized) by

removing recommendations that don't match the current context of the user or adjust the raking provided by the traditional RS.

- Contextual modeling: In this paradigm, contextual information takes part of the modelling technique as part of the rating prediction. The contextual information is used alongside with the matrix user-item as input in recommendation functions expanding the truly multidimensional recommendation.

2.3.5 Combining Multiple Approaches

The combination of multiple Approaches has been studied ensembles the three paradigms for CARS. One option is to develop and combining different method of the same paradigm as shown in Figure 2.4. Adomavicius [AT10] verified this approach by developing a system that merges the results from multiple contextual pre-filters. The same specific context can be generalized in multiple different ways, for example: a context represented by the tuple t_1:(Theatre, Sunday,Girlfriend) can be generalized into t_2:(AnyPlace, Sunday,Friend), t_3:(Theatre, AnyTime,NotAlone), t_n generalizations. Adomavicius use pre-filters for each possible context combining at last the recommendations generated. The recommender aggregator can implement multiple alternatives, could choose the pre-filter with best performance or be used as ensemble of pre-filters.

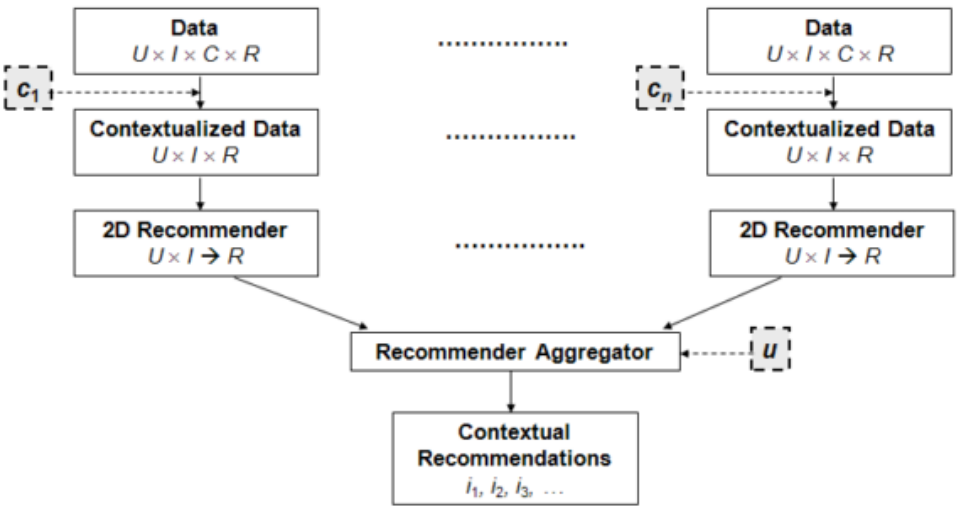


Figure 2.4: Combining multiple pre-filter [AT10]

2.3.6 Issues in Context-Aware Recommender Systems

There are some drawbacks in Contextualization with pre-filter contextual methods when the contextual information restricts too much the data selected. The system may not have enough data to make accurate recommendations with a filter so narrow. Context generalization is used when this problem occurs, creating a hierarchical structure of contextual information to increase the

granularity of the situation[Liu14]. Example: Recommendations for a Friday evening can be generalized into weekend evening which if necessary can be generalized in evening and at the top of the hierarchical is AnyTime. This thought process is represented in Figure 2.5.

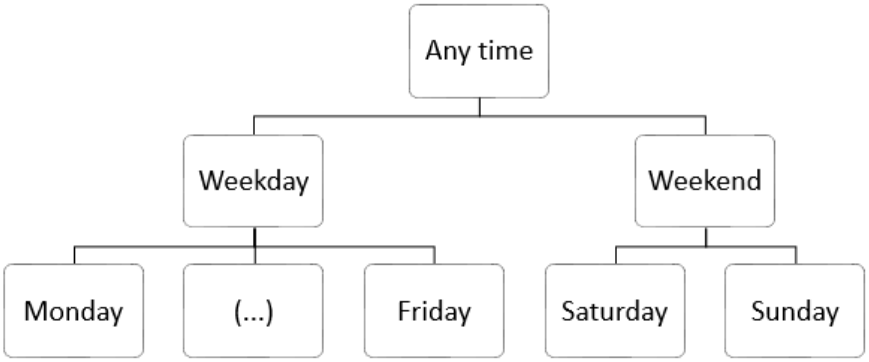


Figure 2.5: Hierarchical structure of contextual information

In most implementation of CARS, the context is initially defined, to be able to generate a hierarchical structure. The problem with this definition is that it cannot match user's definitions, like time for example where the concept of evening can begin later for some individuals. With the limitation of defining all contexts manually, it is used in some cases a more empirical approach that evaluates the predicted performance of the output obtained from each generalization made, choosing the best one. Contextual recommenders must consider two major properties to turn the recommendation methods more flexible [AT10]:

- Complexity: Since CARS is more complex than traditional recommendation systems by not only using the traditional user-item matrix in the process, is important to beware of the various types of contextual information. The complexity allowed on the search made by a user can make the system a lot more complex in relation to the traditional recommenders. Example: A user may seek the top 3 movies to see in the best hours during the weekend with girlfriend.
- Interactivity: To add contextual information to the system, there is a need of asking the user a representation of the context in which he presents himself. Example: asking the user with whom he/she will watch the movie before suggest any recommendation.

2.4 Location aware recommender systems

Location aware recommender systems (LARS) are able to deal with user-related location data, using *user partitioning*, grouping user ratings by the spatial attribute. Another characteristic of LARS is the *travel penalty* favouring the items that are closer in distance. LARS can deal with three types of taxonomy [LSEM12]: spatial ratings for non-spatial items, non-spatial ratings for spatial items and spatial ratings for spatial item.

2.4.1 Spatial Ratings for Non-Spatial Items

Spatial Ratings for Non-spatial Items is based on the tuple (user, user location, item, rating). This approach exploits the user location for the user partitioning method. There must be considered three requirements: Locality, where recommendations are correlated by the preferences of users that are close to the target user; Scalability, the number of users shouldn't be barrier (able to scale up); Influence, the system should be determinate the size of spatial neighbourhood and it's influence.

2.4.2 Non-Spatial Ratings for Spatial Items

Non Spatial Ratings for Spatial Items is based on the tuple (user, item, item location, rating). This approach promotes items that are close to the target user over those at far distance (travel penalty). Travel penalty implies a considerable computational effort to be able to calculate the travel distance to each item. LARS uses employ termination to reduce resource-consuming, terminating the calculation if the systems considered that further search will not produce better recommendations.

2.4.3 Spatial Rating for Spatial Items

Spatial Rating for Spatial Items is based on the tuple (user, user location, item, item location, rating). In order to deal with the two locations, both user partitioning and travel penalty techniques are used on this approach.

2.4.4 Location-Content-Aware Recommender System

Location Content Aware recommender system (LCARS) proposed by Yin et al [YSC+13], offers recommendations having in consideration the localization of the user and his personal interest. LCARS have two component, the offline modeling part (LCA-LDA) designed to define the user's interests and the local preferences, and the online recommendation part that gives the user and local preference as input to the model learnt producing the top-k recommendations. The offline modeling is a location-content-aware probabilistic that aims to imitate the decision making of a user on spatial items [YSC+13]. The model calculates the probability the item be generated given the users interests, and probability that the item would be generated given the spatial attribute of the item merging into a single model. Online recommendation can be represented as a query in the recommendation system, taking two arguments (u,l_u), user u and the location lu,

2.5 Evaluation of Recommendation Tasks

In this Section, it's reviewed the metrics used to evaluate the quality and efficacies of recommendations. The choice of an evaluation metric needs to match the problem and the task of interest [Ana16]. Evaluation metrics of recommendation systems can be categorized into two types[Ote15]: predictive accuracy metrics and classification accuracy metrics.

2.5.1 Recommendation system metrics

Predictive accuracy is one of the most used in recommendation system evaluation metrics. The main idea of this metric is that a recommendation system that predicts rating with accuracy for an item, the items that were given high predictions are most likely relevant items for the user. Predictive accuracy metrics measure the difference between the rating predicted by the system and the rating provided by the user. There are some predictive accuracy metrics that have been helpful in many predictive systems such as "Mean Absolute Error (MAE), Mean Squared Error (MSE), Root Mean Squared Error (RMSE) and Normalized Mean Absolute Error (NMAE)"[Ote15].

1. Mean Absolute Error (MAE): is a quantity used to measure the difference between the recommender's forecasts to the real value [2]. The mean absolute error is given by the formula:

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

2. Root Mean Squared Error (RMSE): is the square root of the mean value of the square of all errors. RMSE is a very popular metric, that is used in numerical predictive evaluations. Compared to similar metrics such as MAE, RMSE amplifies and penalises the existence of large errors. [3]

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

3. Normalized discounted cumulative gain (NDCG): measures the performance of a recommendation system based on the graded relevance of the recommended entities. It varies from 0.0 to 1.0, with 1.0 representing the ideal ranking of the entities. This metric is also commonly used in information retrieval and to evaluate the performance of web search engines.[4]

$$DCG_k = \sum_{i=1}^{k}\frac{2^{rel_i} - 1}{log_2(i+1)]}$$

[2] https://www.kaggle.com/wiki/MeanAbsoluteError [June, 2017]
[3] https://www.kaggle.com/wiki/RootMeanSquaredError [June, 2017]
[4] http://www.ebaytechblog.com/2010/11/10/measuring-search-relevance/ [June, 2017]

Chapter 3

Problem and Solution

In this Chapter the problem being targeted is explained in greater detail and some possible solutions are explained.

3.1 Implementation of the recommender system

In order to respond to both the customer's and business' needs, multiple recommendation system have been developed. In this dissertation the main goal is to create a recommender system capable of responding to at least three scenarios:

- When the customer is planning a shopping trip and a shopping list is being created.
- When the customer is actually shopping in store with indoor navigation being used.
- When the customer is actually shopping in store with indoor navigation not being used.

The system receives as input the User and his current location context if possible. On the first scenario the contextual information can be about the reason's shopping trip, the recommender system is meant to help the user search for product options and advise products that match with the the ones already on the list. On the second and third scenarios, the user is physically inside the store and the recommender must be adapted for two situations: when the users can/wants to give his spacial information, in this case the recommender system can apply concepts mention in 2.4 such as *travel penalty*; when the user doesn't want/can't give is spatial information, but want assistance during the shopping trip.

It may be interesting to detect the differences between results provided by recommender system in the last two scenarios to see how the recommendations change as the information available about the user expands.

3.2 Path finding algortihms

To have the ability of incorporating a path recommendation functionality into the final solution, and at the same time help in defining the user context, the need to study the problem of path finding arose.

Path finding is a well studied problem, where multiple algorithms have demonstrated great performance and flexibility on finding the shortest path [NC]. There are multiple representations for paths, such as matrix or graphs.

3.2.1 Dijkstra's algorithm

When the user receive the recommendations he/she faces a second problem in large stores, what path to take? Dijkstra's algorithm is the gold standard in pathfinding problems [NC]. In his original form, Dijkstra's algorithm finds the shortest path between two nodes in a graph G. In this case, the algorithm must be adapted to have certain nodes classified as "must pass".

The base implementation of Dijkstra is represented by the Pseudocode 1.

Algorithm 1 Find shortest path

```
1:  procedure DIJKSTRA(Graph, source)
2:      for each vertex v in Graph do
3:          dist[v] := infinity
4:          previous[v] := undefined
5:      end for
6:      dist[source] := 0
7:      Q := the set of all nodes in Graph
8:      while Q is not empty: do
9:          u := node in Q with smallest dist[]
10:         remove u from Q
11:         for neighbor v of u do
12:             alt := dist[u] + dist,between(u, v)
13:             if alt < dist[v] then
14:                 dist[v] := alt
15:                 previous[v] := u
16:             end if
17:         end for
18:     end while
19:     return previous[]
20: end procedure
```

The output given by the algorithm is the cost of travel for each node starting in the node s source. The path we want to propose has a particularity, the user doesn't only have a source and a target, but some points in the middle that he must visit (points of interest). To adapt the Dijkstra algorithm to this requirement, apart from the source and end positions, we have a set of positions that he must visit, so that $P = (p_1, p_2, .., p_n)$ where $P \in G$ and G includes the source position together

with the end position. its calculated the distance between all possible pairs, running the algorithm $(|P|)\times(|P| - 1)/2$ times where the dist array holds the distance between all possible pairs.

3.3 Technologies

In this Section it's presented a review about the technologies used during this dissertation. There were two development processes which required different types of technologies: the development of the recommendation system and of the web services for the web application.

3.3.1 MLlib

"MLlib is Apache Spark's scalable machine learning library.". MLlib library offers common learning algorithms and utilities such as classification, regression, clustering and frequent pattern mining.

MLlib excels in two important marks: It's easy to use and can be used in multiple programming languages such as Java, Scala, python and R; MLlib library contains high-quality algorithms that leverage iteration, and can yield better results than systems such as MapReduce which sometimes use one-pass approximations. [1]

3.3.2 LensKit

Lenskit is free and open-source software originally created in GroupLens Research at the University of Minnesota. Lenskit offers numerous tools to research and build recommender applications, two of them used on this dissertation: LensKit Evaluator provides a flexible framework for conducting offline evaluations of recommenders;[2] LensKit Algorithms provides ready-to-use implementations of several recommender algorithms such as Item-based and User-based CF. [3]

3.3.3 Python

Python is a popular programming language that as been raising in popularity between data scientists and machine learning engineers. There are multiple libraries that stand out in Python for data processing purposes such as NumPy, SciPy and Pandas or even for machine learning like scikit-learn, Theano and TensorFlow.

3.3.4 Java

Java can be applied for multiple purposes, being a programming language that is concurrent, class-based, object-oriented, and specifically designed to have as few implementation dependencies as possible. It is intended to let software developers "write once, run anywhere" (WORA), meaning

[1] http://spark.apache.org/mllib/ [June, 2017]
[2] http://lenskit.org/documentation/evaluator/ [June, 2017]
[3] http://lenskit.org/documentation/algorithms/ [June, 2017]

that compiled Java code can run on all platforms that support Java without the need for recompilation[4].

This programming language also allowed the implementation of web services developed in this dissertation.

3.3.5 Web services

A web service is a software system designed to support the exchange and usage of information between machine-to-machine over a network. [5]

In this dissertation will be developed a RESTful web services. In architectural style REST, data and functionality are considered resources and are accessed by using Uniform Resource Identifiers (URIs), typically links on the Web. [6]

There are four principles that encourage good practices in RESTful applications:

1. Resource identification through URI: Resources are represented by URIs, which provide a global addressing space for resource and service discovery. [6]

2. Uniform interface: Resources are manipulated using a fixed set of four create, read, update, delete operations. [6]

3. Self-descriptive messages: Resources are dissociated from their representation so that their content can be accessed in a variety of formats, such as HTML, XML, JSON, and others.[6]

4. Stateful interactions through hyperlinks: Every interaction with a resource is stateless. This means that the request message mus be self-contained. [6]

3.3.6 D3.js

D3.js can be used embedded within an HTML webpage, using pre-built JavaScript functions to select elements, create SVG objects, style them, or add transitions, dynamic effects or tooltips to them. SVG objects can also be styled using CSS. Large datasets can be easily bound to SVG objects using simple D3.js functions to generate rich text or graphic charts and diagrams. The data used can be represented in various formats, being the most common JSON and CSV. Other formats are also supported by writing JavaScript functions adapted to that data format. [7]

[4] https://en.wikipedia.org/wiki/Java_(programming_language) [June, 2017]
[5] https://en.wikipedia.org/wiki/Web_service [June, 2017]
[6] http://docs.oracle.com/javaee/6/tutorial/doc/gijqy.html [June, 2017]
[7] https://en.wikipedia.org/wiki/D3.js [June, 2017]

Chapter 4

Tool for Location Aware Product Recommendation Systems

After a research of datasets that could meet the specifications needed, Ta feng dataset was chosen [1]. Ta feng is a popular dataset used in many recommender systems researches, providing information about its users and products. Even though it is a very complete dataset compared to others available online, some information such as the location of products on store, the user's opinion about the products was missing.

During the development phase, the first step was to understand the dataset and store it in a mysql database. Usually, recommendation algorithms work fine with *csv* or other types of text files, but a database was mandatory to create web-services for the web-application and enhance performance.

After having a database ready for the web application, the next step was to create floor plan to represent a retail store, and generate positions for each product having in consideration its category. A path finding algorithm, between the user's indoor position and the exit store with must-pass points was implemented too.

To present product recommendations to the user's application, three algorithms were implemented: Association Rules, Item-to-Item Collaborative filtering and User-to-User Collaborative filtering.

4.1 Architecture

The developed web application follows a modular approach and various modules interact to achieve full functionality. Later on this Chapter these modules will be reviewed with more detail. The high-level view of the architecture is represented in Figure 4.1 which contains the main components of the system:

[1] http://www.bigdatalab.ac.cn/benchmark/bm/dd?data=Ta-Feng [June, 2017]

- Dataset Parser: To be able to pass the dataset's data as input for two different types of algorithms, this component was developed in Java.
- CF Algorithm: The implementation of the algorithm configured to goal's needs.
- Association Rules algorithm: Implementation of this algorithm and it's configurations
- Database: Store information about products, users, transactions and the output of the three algorithms
- Web services: To facilitate the exchange of information using REST API, between the user and execution of tasks on server side
- Web interface: An implementation of a simple but customizable front-end for the system's user

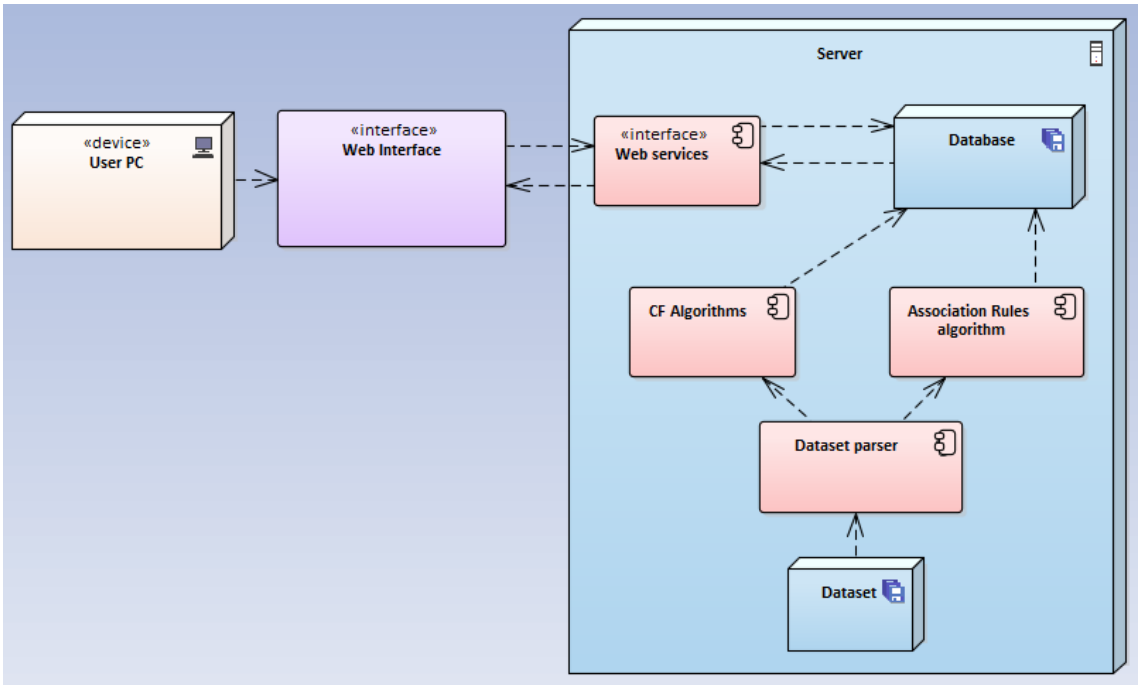


Figure 4.1: System's architecture

4.2 Dataset

Ta Feng is a public grocery shopping dataset provided by ACM RecSys (a premier international forum for the presentation of new research results, systems and techniques in the broad field of recommender systems[2]). The dataset collected user's transactions of 4 months, from November 2000 to February 2001, covering products from food, office supplies to furniture.

Some dataset statistics extracted:

[2] https://recsys.acm.org/recsys17/ [June, 2017]

- 32266 Customers
- 23812 Products
- 2012 Product Categories
- 119578 Transactions
- 817741 Purchases
- Average of 11.85 Items/Category
- Average of 6.84 Items/Transaction

The dataset contains one *csv* file of transactions for each month, making only possible to obtain information about the products sold at least once. Each transaction is represented by one or more rows, each row corresponding to each product bought. The Table 4.1 is an example taken from the dataset.

Table 4.1: Dataset sample

date	customer	age	residence	subclass	product	amount	asset	price
01/01/2001 00:00	141833	F	F	130207	4710105011011	2	44	52
01/01/2001 00:00	1376753	E	E	110217	4710265849066	1	150	129
01/01/2001 00:00	1603071	E	G	100201	4712019100607	1	35	39
01/01/2001 00:00	1738667	E	F	530105	4710168702901	1	94	119
01/01/2001 00:00	2141497	A	B	320407	4710431339148	1	100	159

As we can verify, we have nine attributes present in the dataset:

- date: the transaction date and time, altough time is invalid and useless
- customerId, since it has the shopping history of each user
- age: in a range of { *A,...,J*}, being A < 25, B in {25-29}, C in {30-34}, D in {35-39}, E in {40-44}, F in {45-49}, G in {50-54}, H in {55-59}, I in {60-64}, J > 65
- residence: in a range of { *A,...,F*} representing zip-code areas
- subclass: the category of the productId in that row
- productId: a identifier of one product
- amount: amount bought of that product in that transaction
- asset: this variable is highly correlated with price attribute (99.79%) without representing any meaning and will be ignored
- price: the sales price of transaction's item

4.2.1 Database preparation

It was important for the web application to have the dataset's information organized and optimized to be able to make queries, which csv files do not provide. To optimize and facilitate the information access of the dataset, was created one *mysql* database with three main tables described on Figure 4.2, Product, Subclass, User, Transaction.

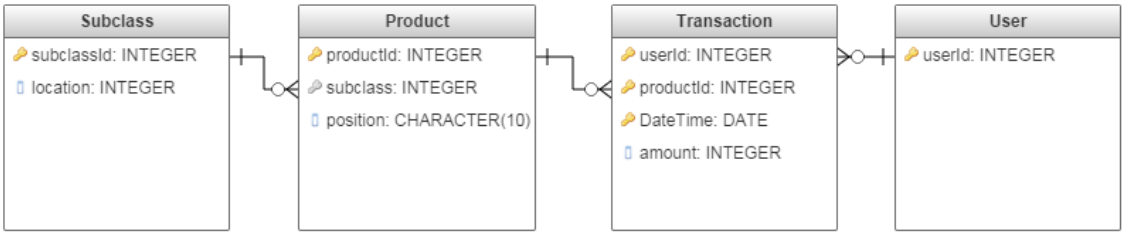


Figure 4.2: Initial database tables

Since the dataset doesn't contain a transaction identifier, to extract one transaction, the transaction's table has a primary *(userId,producId,DateTime)*. For each subclass was generated a shelf number and for each product a position relative to the shelf. The importance of this positions will be explored on Subsection 4.4.1. Another benefit of this database organization is that it facilitates information retrieval about Products and Users using web services.

4.2.1.1 Input for Association Rules

To be able to generate association rules using Apache Spark library, the products of the transactions were organized and grouped in a single line according to the transaction to which they belong. Since the dataset isn't displayed on this way, a Java application was written to gather the transaction information from the table Transactions 4.2, and create a file with the data as shown in the Table 4.2. Each line represent one transaction made, containing all it's products. With this approach, all other informations about the user, amount and date is not considered. The goal of the association rules is to check if there is a correlation about products, "people who bought product(s) $p_{x,..,y}$, tend to buy product p_w with confidence associated.

Table 4.2: Input for association rules

4719111030405 4719111025203 4713045022116 4719111020109
4710189851282 4714381003128 4710515537026
4710105045443 4710339772139 4710063341090
9310022733406 20570637 4901422018931
20570552 20470340 20563776 20460631

4.2.1.2 Input for Collaborative Filtering

To be able to generate recommendations based on Collaborative Filtering, it was necessary to create an User-Item matrix to pass as input. This matrix represents the rating that the user gave to the items he experienced. This rating was not present in the dataset, so it was necessary to calculate a pseudo-rating (affinity). The rating calculated is given by the number of times the product was present on the target user's transaction, divided by the number of all transaction of the target user. This was performed by a sql query that converted the rating output into a value ranged between $[1,5]$ and then exported as a *csv* file represented in the Table 4.3.

Table 4.3: Input for collaborative filtering algorithms

userId	productId	rating
2159973	4711863180070	3.4
2159973	4711271000014	1.8
2159973	4710063151149	3.4
2159973	4711128778875	2.6
16766	4714050000328	1.8
16766	4714981010038	2.6
16766	4710291112172	1.8

4.3 Product Recommendations

The first step to develop a product recommendation system, is to choose the algorithm that powers the recommendations. Facing this choice we need to analyze the data and decide what kind of based-recommendations we want to offer.

Since the dataset lacks in meta-data about the products and we had transaction data, Association rules was seen like a good choice and answered the question "Clients who brought item(s) *x,..y*, tends to buy item *w*" with confidence associated.

After seeing the results from Association Rules algorithm, it was possible to observe that the recommendation system couldn't recommend products in a lot of cases, due to lack of support that the system calculated for some itemsets at the "Antecedent side" of association rules. To be able to recommend more items, it was decided to use collaborative filtering based algorithms to produce extra recommendations targeted to the user ending up (hopefully) being part of all recommender systems.

At last, was merged all recommendations output from the three algorithms, the system can recommend now by looking at the user's basket (Association Rules) and by analyzing the user's history (Collaborative Filtering).

Represented in the Figure 4.3 the recommendation generation process, shows the merge of two processes:

- On the left side, the process begins with data information about the Users (U), Items (I) and the affinity between them (R), this information is passed as input to two different recommendation algorithms: Item-based and User-based. The Top_{20} recommendations is generated for each user from the two algorithms output, then, taking into consideration the user's location and his basket's products, the recommendations are then sorted by recommendation score and weight path and presented to the user.

- On the right side, the process begins with transaction data (T) which is passed as input to association rules algorithm. Then, if any of the basket's products is present on any rule's antecedent side, these rules are sorted by their confidence and weight path and presented to the user.

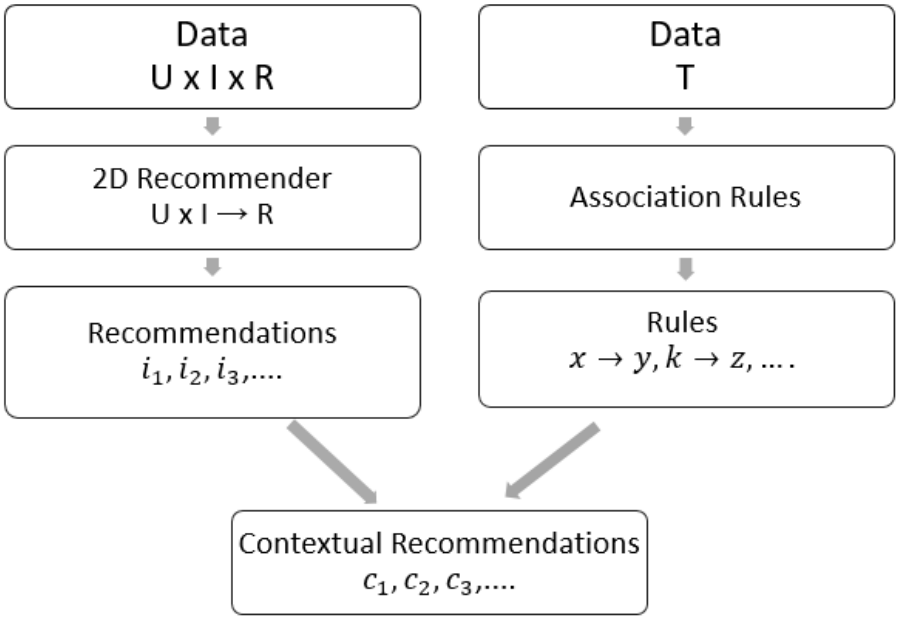


Figure 4.3: Process for generating recommendations

The recommendation's list of contextual recommendations presented to the user is sorted in ascending order by the percentage difference value. This means that for each recommendation is compared with optimal recommendation calculated through the following formula [3]:

$$\frac{|V_1 - V_2|}{\frac{(V_1+V_2)}{2}} \times 100$$

This formula gives the percent difference between two positive numbers greater than 0. Applying this formula to the problem of this dissertation work, the value of the formula is given by

[3] https://www.calculatorsoup.com/calculators/algebra/percent-difference-calculator.php [June, 2017]

the weighted sum of the percentage difference between the ideal distance and the required distance and between the maximum confidence value and the confidence value of the product under analysis:

$$\frac{|CurrentPath - NecessaryPath|}{\frac{(CurrentPath+NecessaryPath)}{2}} \times RecPath + \frac{|MaxAffinity - ItemAffinity|}{\frac{(MaxAffinity+ItemAffinity)}{2}} \times RecConf$$

4.3.1 Association rules

Association rules is as well known approach in transaction data. After some data preparation detailed in 4.2.1.1, Apache Spark's Machine Learning Library (MLLib) implementation was used to generate rules.

This library uses another algorithm to produce frequent itemsets with support associated for memory optimization purposes (FPGrowth). Unlike Apriori-like algorithms, FP-Growth, where FP stands for frequent pattern, uses a suffix tree (FP-tree) to encode transactions reducing the FPGrowth's filters the transaction data, given as output frequent itemsets, used afterwards as input by the association rules model.

This implementation allows to define the minimum support and confidence, used in this case 0.0001 for support (at least 5 transactions), and 0.1 for confidence. This model produced 15492 association rules out of 23812 items.

Some results from Association rules:

- There are associations rules with max confidence (1)
- The best seller product was bought 5475 times (support)
- From all 15492 associations rules generated, there are 5968 distinct values for the antecedent side

4.3.2 Collaborative Filtering

Collaborative filtering, reviewed in 2.1.1, is an approach that revealed great success when the lack of information about the products doesn't allow the use of Content-based methods and when the scalability is a must.

Collaborative filtering requires a user-item matrix as input, so, after the data preparation process mention in 4.2.1.2 a matrix with 32266×23812 of dimension was passed as such.

The Collaborative filtering algorithms were implemented using LensKit's library 3.3.2. Both algorithms tried to produced *Top20* recommendations directed for each user, returning 1049652 recommendations in total.

Each recommendation has one score associated, that has a direct proportion with the probability of the product being appropriated for the user. From 1049652 recommendations generated, only 10 recommendations were present on the same user's *Top20* .

4.3.2.1 Item-based CF

The main idea of Item-based recommender is to compute similarities between items, typically based on the users that have rated them and generate recommendations for each user, similar to the products that the user liked.

The configurations used for the recommender system were the ones suggested by the Lenskit group [4], using 20 neighbours for each prediction (default value).

Item-Item produced 645052 recommendations, since the dataset provides 32266 users, it was expected to produce 645320 recommendations (*Top20* for each user). The user's that couldn't make *Top20* recommendations, suffers from the same problem, only bought one product, making it difficult for the system to complete the item similarity matrix for the target user.

4.3.2.2 User-based CF

The configurations used for the recommender system for comparison purposes were the same as the Item-based algorithm.

User-based algorithm produced 404600 recommendations, since the dataset provides 32266 users, it was expected to produce 645320 recommendations (*Top20* for each user). The users that couldn't make *Top20* recommendations, suffers from a different problem compared to Item-based, the affinity extracted from these users were all the same for each product they consumed, making it difficult for the system to compute similarities between user's.

4.3.3 Hybrid Approach

After the implementation of the two Collaborative Filtering algorithms, it was tested if the combination of the two scores with different weight would bring benefit results when predicting the affinity between the users and products. For each score predicted, the formulas and conditions would be:

$$Score_h = k \times Score_i + y \times Score_u$$

$$k + y = 1$$

$$k, y >= 0.1$$

$$k, y < 1$$

Being $Score_h$ the hybrid score, $Score_i$ the item-based score, $Score_u$ the user-based score, k and y the weight for each collaborative filtering algorithm. This approach was not that successful, detailed results are demonstrated in 5.3.

[4] http://lenskit.org/documentation/algorithms/item-item/ [June, 2017]

4.3.4 Merging recommendations

After the implementation of the three algorithms previously introduced, it would be beneficial to see if the *TopN* recommendations presented to the target user contained a combination of the output produced by the three algorithms.

Since Collaborative filtering User-based and Item-based try to predict the affinity that the users would give to new products but with different approaches it can be compared very easily, sorting the output of those two algorithms by their affinity (and later on, with path weight).

The main issue is comparing the CF's output with the association rules extracted from the transactions. Given the low number of association rules generated compared to the quantity of products, and the generalization of this rules/recommendations, it was decided that if any product of the user's basket is an antecedent of any rules it will be included on *TopN* recommendations present to the user.

4.4 Location Aware Recommender

In this Section it is described the concept of the implemented recommender and why it can be applied to multiple scenarios. This recommender system has also a path finding algorithm integrated using an extend of the Dijkstra algorithm 3.2.1.

4.4.1 Concept

The main goal of the recommender system developed is to generate different and more personalized recommendations taking into consideration the user's indoor localization. This means that the recommendations present to the user should not be generic recommendations and to distinguish from traditional recommender systems the travel path of the user is also suggested and had this in mind.

To simulate multiple shopping trips a store's floor plan was generated with 35 shelves present in Figure 4.4. Each shelve is represented by an unique colour, and the store's exit/entry on the bottom left without any colour.

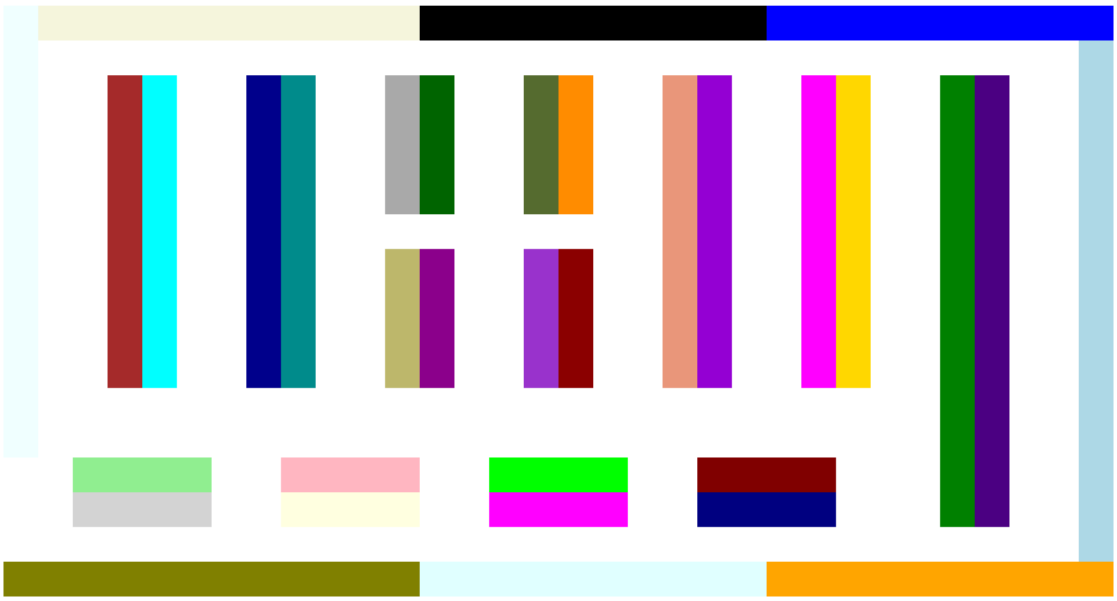


Figure 4.4: Store Floor Plan

After plan was defined, it was necessary to distribute all products on the shelves. To accomplish this task, all categories (subclass column on dataset) were allocated among the shelves. The user can go to one shelve in different positions, to specify the position of the products along the shelf it was generated for each product, the position that the user must be to catch that product, for one shelve all positions around are valid. For a better representation, Figure 4.5 shows a grid view of the floor map, showing all positions that the user can take in the store and around the shelves.

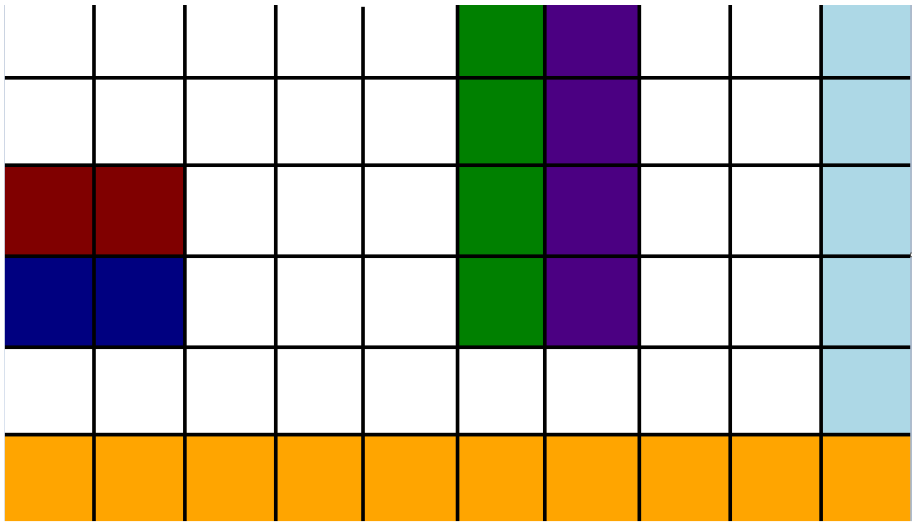


Figure 4.5: Grid View of Store Floor Plan

To be able to use path finding algorithms, the store was represented in graph form. The nodes were all the positions that the user can be in the store (blank squares on Figure 4.5), giving 1

unit of weight between nodes which are in cardinal points direction (north, east, south, and west), and $\sqrt{1^2+1^2}$ (Pythagorean theorem) units between nodes which are in intercardinal directions (northeast, southeast, southwest and northwest).

4.4.2 Path Finding

The shortest path problem is a well known problem, where the goal is to find a path between two vertices (nodes) in a graph such that the sum of weights of the edge's path is minimized.

As reviewed in 3.2.1, *Dijkstra* is one of the most popular algorithm to overcome this problem, but in this project we face another type of problem, the path is not only between two points, it has also N must pass points. The end position is always the store's exit position, the first position is the current user's location and the must pass points are product's positions which the user's has in his basket or manual inserted positions.

This problems isn't a Travelling salesman problem (TSP) [Cor13] because the client doesn't have to visit every shelf. If we take into consideration the dataset's history purchases made, each transactions has as 7 items on average, and given 7! is rather small (5040), was used permutations of only the must pass nodes using *Dijkstra's* algorithm to find the shortest distance between each pair of vertices.

The solution passed then from generating permutations between the must pass nodes, it is given to the webservice a list of nodes e.g: $[init, a, b, c, exit]$, and calculated the full weight of each permutation, in this case the server would calculate the path weigh of this permutations:

- $[init -> a -> b -> c -> exit]$
- $[init -> a -> c -> b -> exit]$
- $[init -> b -> a -> c -> exit]$
- $[init -> b -> c -> a -> exit]$
- $[init -> c -> a -> b -> exit]$
- $[init -> c -> b -> a -> exit]$

It returns the permutation with lowest weight, alongside with all other nodes between, to be able to draw on front-end, exemplified on Figure 4.6.

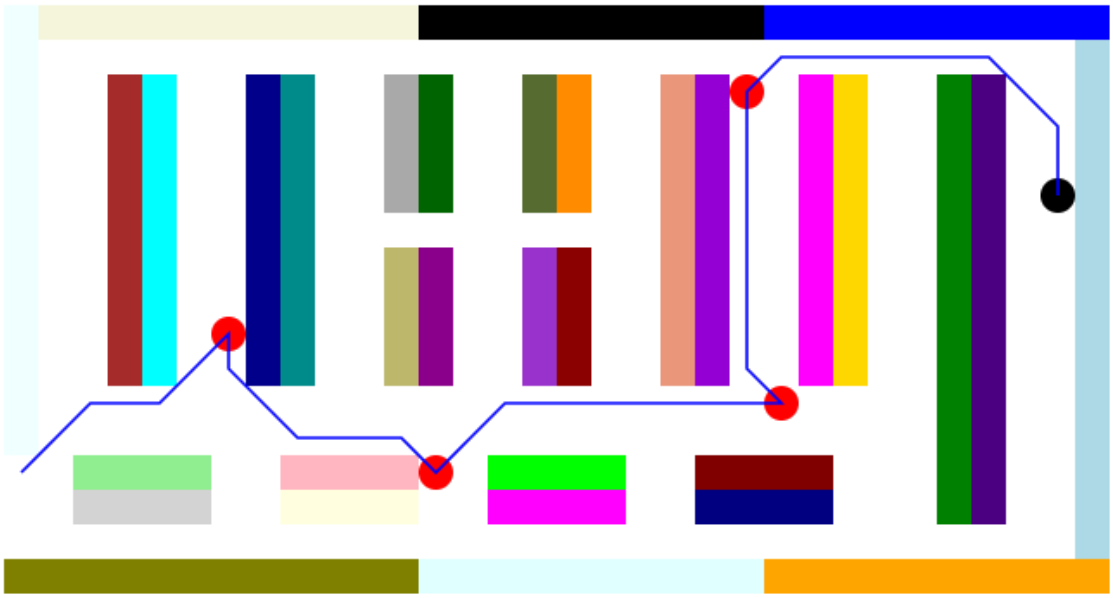


Figure 4.6: Example of Path found

4.4.3 GUI

In order to build a good GUI with good drawing tool for the store plan floor, it was chosen D3.js library 3.3.6. D3.js allowed to build build a plan floor based on a graph, and made it also possible to visualize the user's and product's position and the suggested path. Other library's were used too, given the large number of products, it was unpratical to provide all products in one simple html select, so Select2 [5] was used to overcome this problem. Other library used was Bootstrap [6] to have a simple but good looking html elements.

[5] https://select2.github.io/ [June, 2017]
[6] http://getbootstrap.com/ [June, 2017]

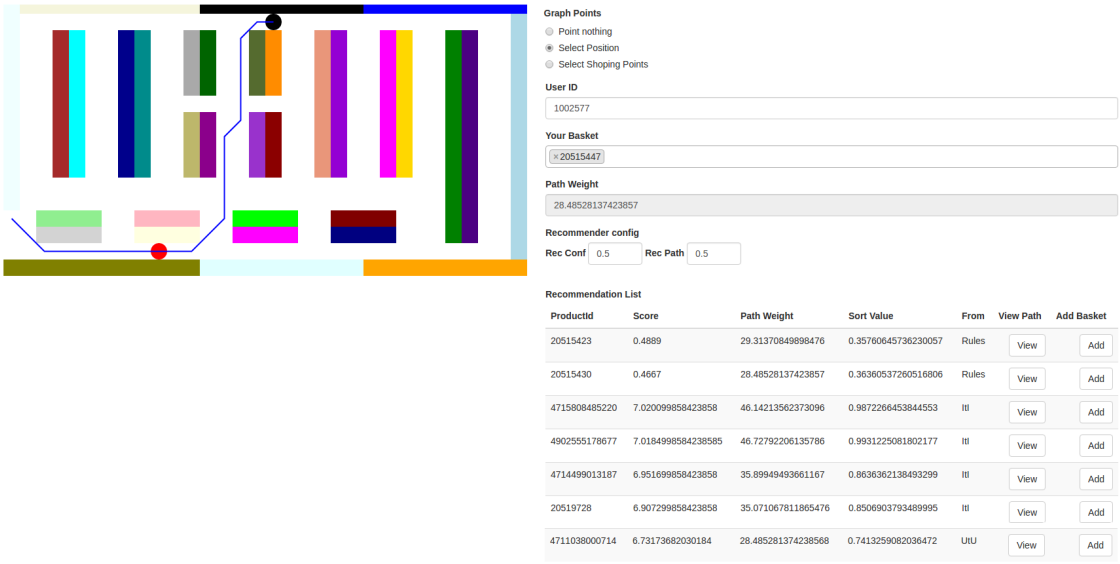


Figure 4.7: Full Page

The GUI as shown in Figure fig:fullpage has two parts, the Floor plan and the user's form to test and use the system:

- Store Floor Plan of the store, where the user can define his position and visualize products location and recommended paths.

- User's form for simulation purposes, either simulate a client experience, change recommender settings or both.

4.4.3.1 Store Floor Plan

To be able to draw the store floor plan with a simple for loop, the graph was converted onto a 2d array. The GUI user doesn't see the grid view previously demonstrated 4.5, but is able to draw a point and subsquent path in each square cell. The user can interact with this maps in only two ways: Change the client's current position or add must pass points (points of interest), representing the client's need of going into some shelve, without knowing exactly if he is going to bought any product around that area.

The front-end calls the web services for path finding, the current position was marked, and the user has products in the basket or points of interest marked or both. There are two main stages of the map interface, the one that the user has the current suggested path as shown in Figure 4.8, taking into consideration his basket's products and the path that the user's would find if he chose a suggested product as shown in Figure 4.9.

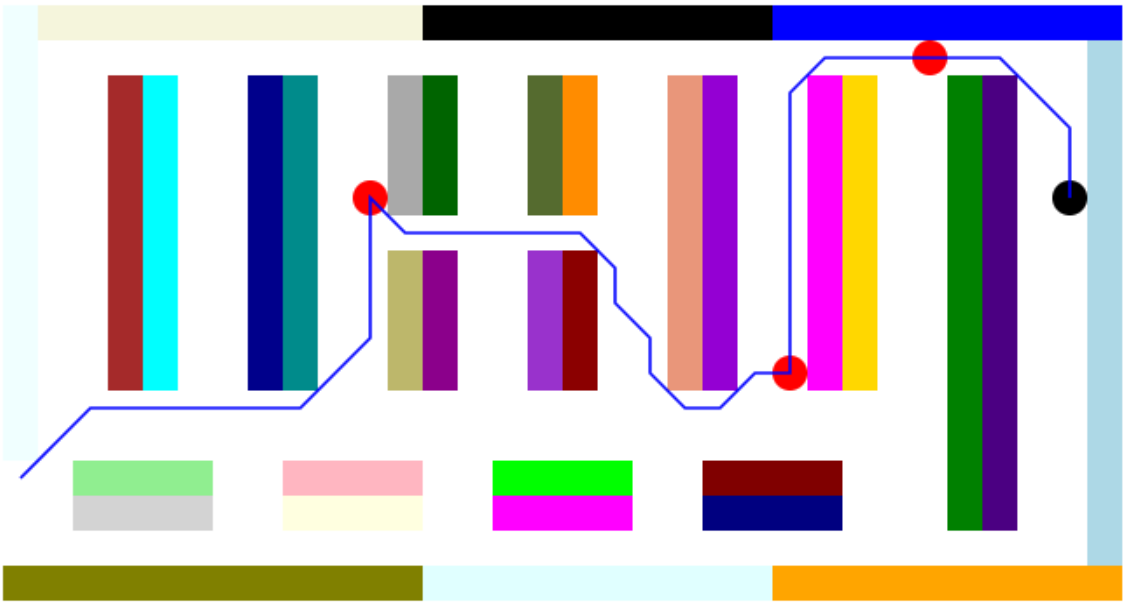


Figure 4.8: Recommended Path

The user's position is represented as black circle, points of interest/basket's products as red circles and the recommended path marked as blue.

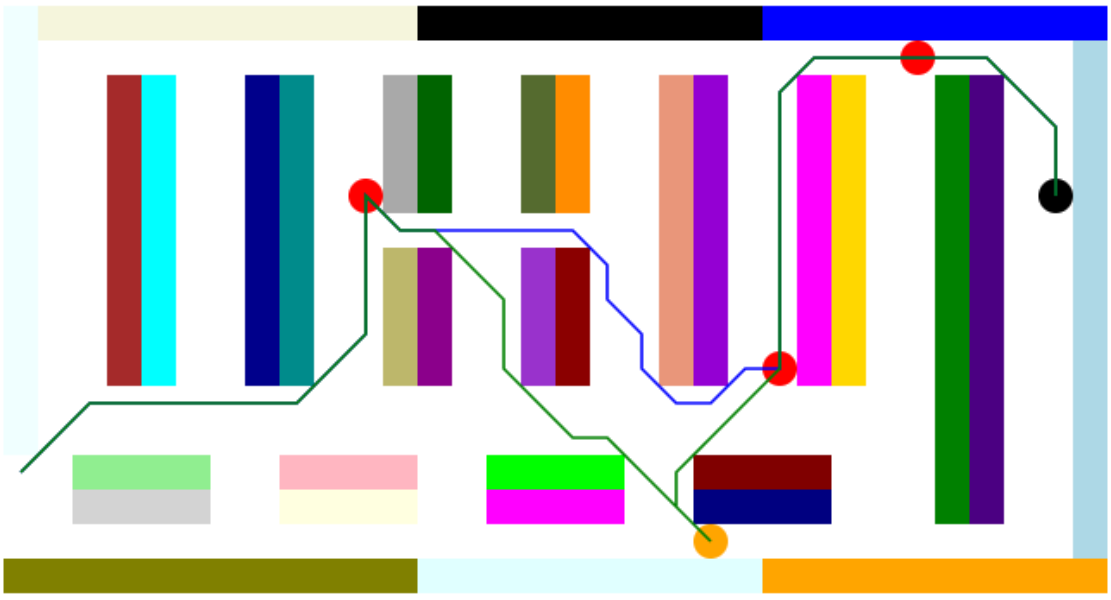


Figure 4.9: Path with suggested Item

The recommended path with a suggested product has the same elements, but with the suggested product location is marked as yellow, and the modified path as green. With this view it is

possible to compare the two recommended paths, making it easy to see the travel changes that the client would have to make.

4.4.3.2 User's form

For simulation purposes, the user using this form can both simulate a present user on a typical case or change the basket contents, the user start and end positions and the recommender systems configurations.

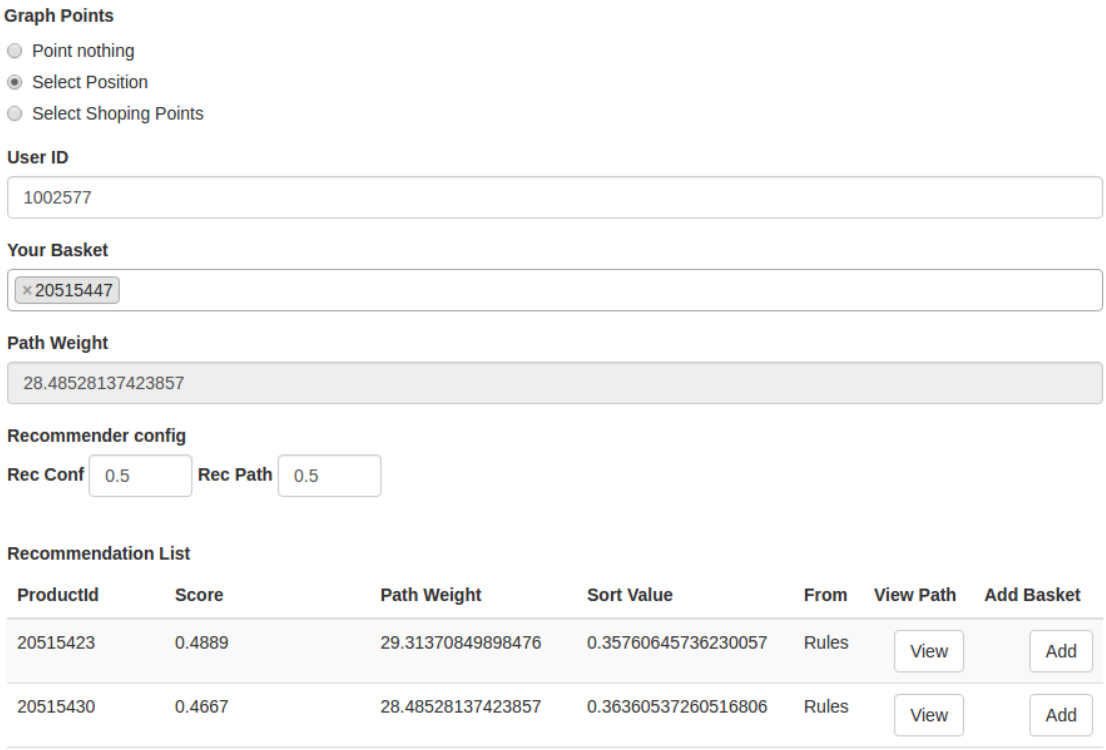


Figure 4.10: Page's Form

As present on Figure 4.10, the user's form is divided in multiple features that can be manipulated:

- Change de position of the user on the floor plan store and his points of interest
- Change the user's id for which the recommender system will generate recommendations
- Add or remove products from the user's basket
- Verify the weight value of the recommended path
- Change two settings of the recommender system to order the recommendations in a particular way, either mostly by the extra path gain or recommendation's affinity, the sum of these value most be 1

- Check the recommendations list generated by the recommender system when at least two inputs are present: the user's location and at least one product on the basket
- View the path change that a recommendation would make and add directly that recommendation to the user's basket

Chapter 5

Tests and Results

In this Chapter the results are analyzed, validity and actual relevance of the recommendations and of the recommended paths.

5.1 Recommendation Accuracy

To be able to assure the recommender systems quality, the traditional recommender algorithms present on the system were tested. The Evaluator system, was made using the Lenskit Evaluator framework [1], that allows the usage of cross-fold validation. This evaluator calculates the recommender performance metrics reviewed in 2.5.1 and others that will be analysed during this Section.

5.1.1 Association Rules

When using association rules, the generated rules generally reflect tends present in the dataset. Which means, association rules is not directed to a target user, rather reviewing trends of the client's majority habits. These majorities can be manage giving as input minimum support for the first phase (generation of frequent itemsets) and minimum confidence for the second phase (generation of association rules from itemsets). This means that to be able to detect minority trends, these two values must be low. During this recommendation system development, the minimum support was 0.0001, at least 12 transactions, and the minimum support was 0.1, on average the association rules generated has 0.51247 of support.

On Table 5.1 some generated association rules are presented. As we can verify, frequent itemsets can be more than just one product (on antecedent side), but only one product is present on consequent side. For each association rule, was calculated its confidence, the number of transactions that the Antecedent's itemset was present (freqAntec) and at last, the number of transactions which the antecedent and consequent itemsets were present together (freqUnion).

[1] http://lenskit.org/documentation/evaluator/ [June, 2017]

Table 5.1: Association rules output

consequent	antecedent	conf	freqAntec	freqUnion
20508982	20548636	0.6471	17	11
20515423	20515447	0.4889	45	22
20515423	20515447, 20515430	0.6667	21	14
20515430	20515447	0.4667	45	21
20515430	20515447, 20515423	0.6364	22	14
20515447	20515430, 20515423	0.6087	23	14
20555092	20456245	0.4412	68	30
20571023	20571122	0.4894	47	23

5.1.2 User-based model evaluation

Using *k-fold* validation was calculated the metrics previously reviewed 2.5.1. Thanks to Lenskit Evaluator, it was also possible to verify this metrics by User (MAE and RMSE). The *K-Fold* validation's results present on Table 5.2 can be used to extract some knowledge:

- User-based model requires little time to build, but can take a considerable time to generate recommendations (Test time)
- Could produce scores for all data test's products (NGood)
- Overall good results on forecast metrics (MAE and RMSE) taking into consideration that the affinity given to the recommender system was simulated, not having on the dataset it's real value
- The recommender system can produce relevant recommendations based on the graded relevance of the recommended entities (nDCG)

Table 5.2: User-to-User collaborative filtering evaluation

Part	BuildT	TestT	NGood	MAE	RMSE	nDCG
0	406	69586	51302	0.8987	0.9555	0.9944
1	435	70576	50760	0.9099	0.9652	0.9945
2	444	68644	50421	0.9386	0.992	0.9950
3	527	70365	51030	0.9053	0.960b	0.9948
4	478	69989	51149	0.9072	0.9631	0.9947

In order to verify if the k-validation was trust wordy and measure the metrics in a simple way, was generated Table 5.3 where we can see the standard deviation is very low.

Table 5.3: Average and standard deviation of User-to-User evaluation

	MAE	RMSE	nDCG
Average	0.9119	0.9676	0.9947
Standard deviation	0.0138	0.0135	0.0002

5.1.3 Item-based model evaluation

To test Item-based model, it was used the same folds of User-based, with the same partitions. The results which are present in Table 5.4 reveal some important points:

- Unlike the User-based model, the build time requires more time than the test time
- Could also produce scores for all data test's items (nGood)
- Item-based produce scores with more error comparing with User-based model, but has good results overall
- The recommender system could also produce relevant recommendations based on the graded relevance of the recommended entities (nDCG)

Table 5.4: Item-to-Item collaborative filtering evaluation

Part	BuildT	TestT	NGood	MAE	RMSE	nDCG
0	26632	4792	51302	0.9873	1.0882	0.9835
1	26764	5216	50760	1.0012	1.0969	0.9837
2	37356	15432	50421	1.0277	1.1220	0.9840
3	28434	5670	51030	0.9962	1.0943	0.9835
4	26225	5466	51149	0.9995	1.0973	0.9841

The results were also compiled into other Table 5.5, where we can verify the low standard deviation of this tests and it's average.

Table 5.5: Average and standard deviation of Item-to-Item evaluation

	MAE	RMSE	nDCG
Average	1.0024	1.0998	0.9838
Standard deviation	0.0135	0.0116	0.0003

5.1.4 Hybrid-based evaluation

Since the user-based and item-based model were implemented and tested, it was opportune to ask what the results could be if the scores were combined from the two models and if could produce less error and better results overall. Using the *k-fold* validation results, it was possible to verify the predictions made, exemplified in Table 5.6.

It was calculated then a new hybrid prediction based on these two models and the same RMSE and MAE metrics were calculated.

Table 5.6: Hybrid predictions sample

User	Item	Rating	IPrediction	UPrediction	HPrediction
7795	20522285	2	2.386263862	2.192564494	2.211934431
7795	28851030212	3	2.81281304	2.162992595	2.22797464
7795	50000301829	3	2.275757331	3.505731295	3.382733899
7795	4710126090682	2	1.784241277	2.122721882	2.088873822
7795	4710466103073	2	2.211171453	2.096761272	2.10820229
7795	4710498234202	3	2.357859723	2.325343938	2.328595517
7795	4713888701520	2	2.332	2.184763441	2.199487097

Using solver equation to reduce the error present in the mention metrics by the two arguments explained in 4.3.3 it was verified the following results shown in Table 5.7. The results were below expectations, and the best solution gave more importance to the User model (0.9 weight) and had worse evaluation than the User model.

Table 5.7: Hybrid evaluation

Item	User	RMSE	MAE
0.1	0.9	1.105830455	0.603788458

5.2 Recommended paths

The path finding algorithm implemented was tested in different scenarios (floor plans) and with different positions for different clients and products. It had a good performance overall, being always able to generate one of the shortest path, since in multiple stages the algorithm can find multiple shortest paths with the same weight.

If there are multiple shortest paths with the same weight, the system will choose one of them, randomly.

The figures 5.1 and 5.2 presents the solutions found in two scenarios with different floor plan and positions, in both cases there are more than one solution but with the same weight and usually visiting the positions on the same order.

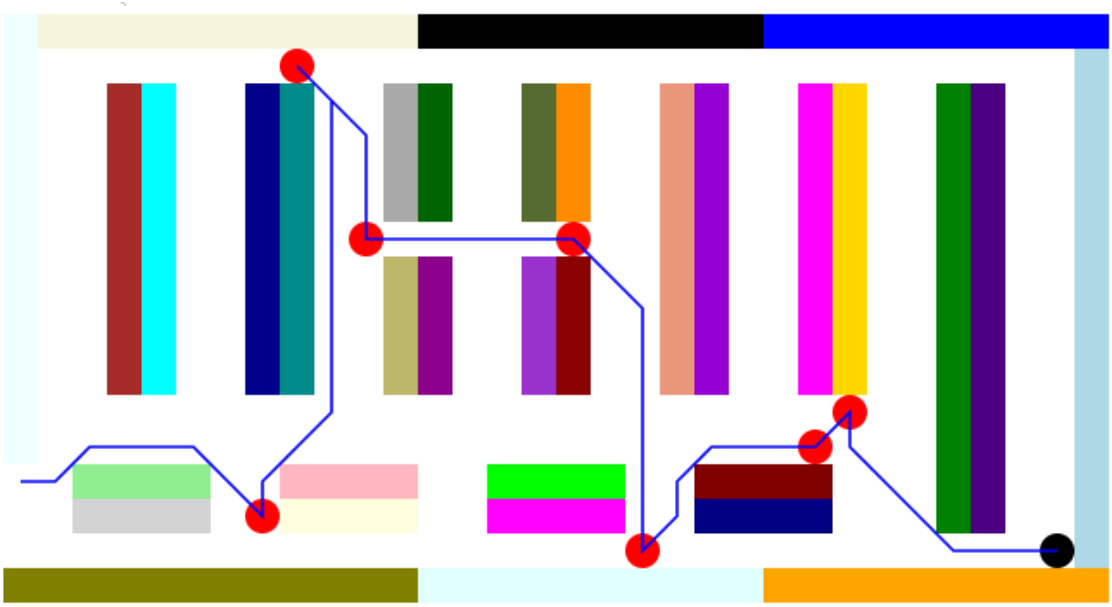


Figure 5.1: Path generated Plan Floor 1

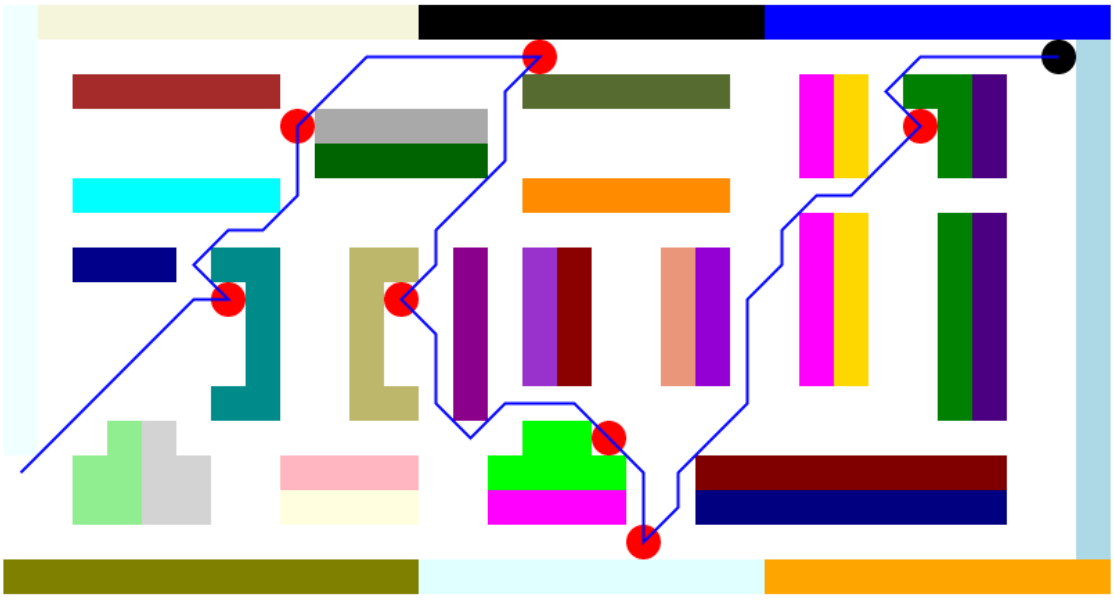


Figure 5.2: Path generated Plan Floor 2

5.3 Results

Along this Section will be present some results from the system implemented which can be fully consulted in Appendix A. In order to validate the main idea of this recommender system, recommend products according to the user's indoor position the scenarios will have characteristics that

meet this purposes summarized on the Table 5.8:

- The target user will remain the same for comparison purposes and so are his basket.
- It will be demonstrated in 2 different store's floor plan scenarios.
- The recommender system settings will have 3 stages: when the user's prefers low weight path rather than optimal recommendations based on his likes and the opposite situation; When the recommender system gives the same weight for both configurations, the weight path and optimal recommendations.

Table 5.8: Result's structure

UserId	Basket's Products	Floor plan	Recommender's Config	Results
1001679	20515447,10742202211,4103040328120	Figure 5.3	Conf: 0.1 and Path: 0.9	Table 5.9
			Conf: 0.5 and Path: 0.5	Table 5.11
			Conf: 0.9 and Path: 0.1	Table 5.10
		Figure 5.4	Conf: 0.1 and Path: 0.9	Table 5.12
			Conf: 0.5 and Path: 0.5	Table 5.14
			Conf: 0.9 and Path: 0.1	Table 5.13

5.3.1 Store's floor plan

In accordance with the goal of this dissertation, was developed two different Store's floor plans, making it easy to validate and view the path finding algorithm output and the recommendation's ordering. In both floor plans it's distributed the products in the same way, this is, the distribution based on the product's subclass is the same. While the product's location is different because of store organization it is present on the same shelve color as we can verify in both figures 5.3 and 5.4.

The performance of path finding algorithm is essential to this system, not only it is used as recommendation to the user, for better indoor orientation and time saving it is used as part of the system to calculate and order the recommendation lists recommended to the user.

On Store's floor plan 1 present in Figure 5.3 it is shown a typical organization of a retail store and was the most used floor plan for developing purposes and test. The second Store's floor plan present in Figure 5.4 was only created to verify the path finding algorithm with a kind of messy organization and testing overall.

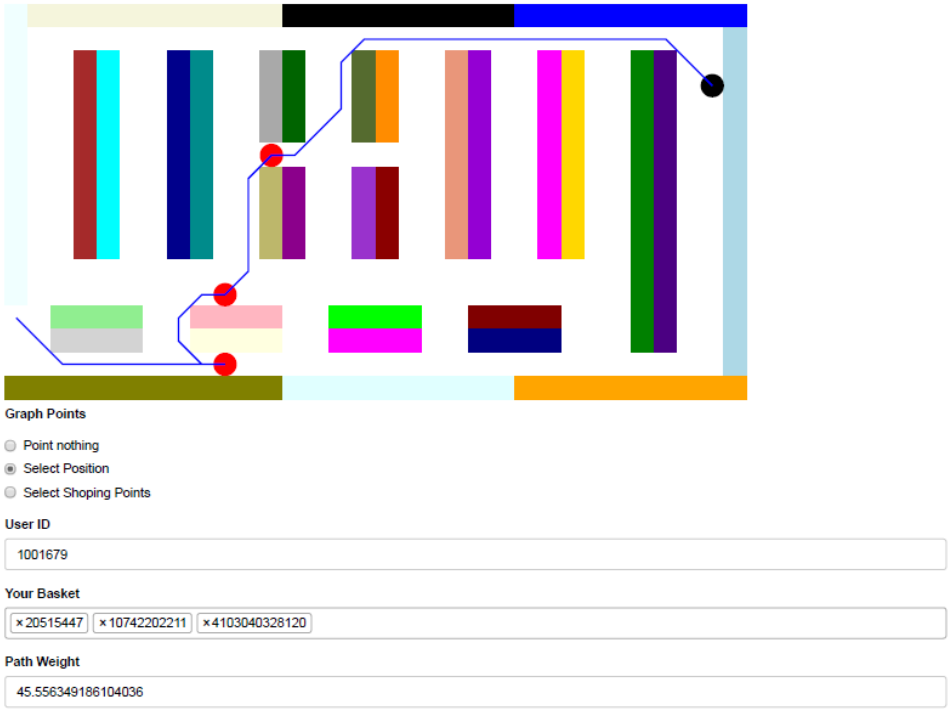


Figure 5.3: Store's floor plan nº1

Figure 5.4: Store's floor plan nº2

5.3.2 Recommendation system's configuration

One of the main features of this application is the adaptation of the recommender system to different scenarios. To be able to adapt the system to the different scenarios it receives two inputs: *RecConf* and *RecPath*, with these two inputs the systems meets the requirements initially proposed:

- If the user's doesn't have limit time to navigate in the store, the system can recommend as a 2D traditional recommender system and with the help of association rules previously generated.

- If the user's has limit time to navigate in the store, the system can recommend relevant products without often having to change the weight user's path.

- The default recommender configuration is 50%-50% for both parameters, where the system balance the time spent and the recommendation score or confidence.

For the main Store floor plan 5.3, was extracted three scenarios present on Tables 5.9, 5.10 and 5.11. As we can verify the minimum weight path is ≈ 45.5563, so, in all these three scenarios the suggested product will make the weight path stay equal or increase.

It is common that the weight doesn't change that much on some recommendations, since it is suggested sometimes products of the same category as the basket's products (in association rules output for example), so the product will be on the same shelf.

In this three scenarios it is noticeable the different ordering depending on the parameters given to the system and its priority. It is present also one column called "From" to see from which algorithm the recommendation was generated for testing purposes.

Table 5.9: Results with Conf:10%, Path:90% in Store floor plan nº1 5.3

ProductId	Score	Path Weight	Sort Value	From
20515423	0.4889	45.55634919	0.068654712	Rules
20515430	0.4667	47.55634919	0.1113839	Rules
4710091110491	6.966410256	45.55634919	0.013812126	UtU
4710734001155	6.905965517	45.55634919	0.01467915	UtU
4710177012060	6.825666667	45.55634919	0.015841896	UtU
44738072342	6.920714286	46.38477631	0.03068568	UtU
4712425010392	6.896952381	46.38477631	0.031027773	UtU
4710020150017	7.411813953	47.55634919	0.04629575	UtU
15000021283	6.980902857	47.55634919	0.052268109	ItI
4710088414687	7.078878437	49.21320344	0.081673624	UtU

Table 5.10: Results with Conf:90%, Path:10% in Store floor plan nº1 5.3

ProductId	Score	Path Weight	Sort Value	From
20515423	0.4889	45.55634919	0.617892404	Rules
20515430	0.4667	47.55634919	0.65878554	Rules
4710020150017	7.411813953	47.55634919	0.072992188	UtU
4710088414687	7.078878437	49.21320344	0.117673736	UtU
4710091110491	6.966410256	45.55634919	0.124309137	UtU
15000021283	6.980902857	47.55634919	0.12674342	ItI
44738072342	6.920714286	46.38477631	0.132004581	UtU
4710734001155	6.905965517	45.55634919	0.132112346	UtU
4712425010392	6.896952381	46.38477631	0.135083418	UtU
4710177012060	6.825666667	45.55634919	0.142577062	UtU

Table 5.11: Results with Conf:50%, Path:50% in Store floor plan nº1 5.3

ProductId	Score	Path Weight	Sort Value	From
20515423	0.4889	45.55634919	0.343273558	Rules
20515430	0.4667	47.55634919	0.38508472	Rules
4710020150017	7.411813953	47.55634919	0.059643969	UtU
4710091110491	6.966410256	45.55634919	0.069060631	UtU
4710734001155	6.905965517	45.55634919	0.073395748	UtU
4710177012060	6.825666667	45.55634919	0.079209479	UtU
44738072342	6.920714286	46.38477631	0.08134513	UtU
4712425010392	6.896952381	46.38477631	0.083055595	UtU
15000021283	6.980902857	47.55634919	0.089505764	ItI
4710088414687	7.078878437	49.21320344	0.09967368	UtU

For the second Store floor plan generated 5.4, was also extracted three scenarios present on Tables 5.12, 5.13 and 5.14. As we can verify the minimum weight path is ≈ 43.3137, therefore, in all these three scenarios the suggested product will make the weight path stay equal or increase.

It is possible to make the same conclusions as in the previous case, and if we compared the same configurations with the previously example, we can verify that depends only with the weight path that the user's would had if he choose one of the recommendations.

Table 5.12: Results with Conf:10%, Path:90% in Store floor plan nº2 5.4

ProductId	Score	Path Weight	Sort Value	From
20515423	0.4889	43.3137085	0.068654712	Rules
20515430	0.4667	43.3137085	0.072721075	Rules
15000021283	6.980902857	43.3137085	0.013605283	ItI
4710091110491	6.966410256	43.3137085	0.013812126	UtU
44738072342	6.920714286	43.55634919	0.019494606	UtU
4713045026183	6.907111111	43.55634919	0.019690313	UtU
4710734001155	6.905965517	43.55634919	0.019706811	UtU
4710177012060	6.825666667	44.97056275	0.049622971	UtU
4710020150017	7.411813953	47.79898987	0.096243037	UtU
4710088414687	7.078878437	50.14213562	0.143735836	UtU

Table 5.13: Results with Conf:90%, Path:10% in Store floor plan nº2 5.4

ProductId	Score	Path Weight	Sort Value	From
20515423	0.4889	43.3137085	0.617892404	Rules
20515430	0.4667	43.3137085	0.654489671	Rules
4710020150017	7.411813953	47.79898987	0.078541887	UtU
15000021283	6.980902857	43.3137085	0.12244755	ItI
4710091110491	6.966410256	43.3137085	0.124309137	UtU
4710088414687	7.078878437	50.14213562	0.124569538	UtU
44738072342	6.920714286	43.55634919	0.130761128	UtU
4713045026183	6.907111111	43.55634919	0.132522494	UtU
4710734001155	6.905965517	43.55634919	0.132670975	UtU
4710177012060	6.825666667	44.97056275	0.146330515	UtU

Table 5.14: Results with Conf:50%, Path:50% in Store floor plan nº2 5.4

ProductId	Score	Path Weight	Sort Value	From
20515423	0.4889	43.3137085	0.343273558	Rules
20515430	0.4667	43.3137085	0.363605373	Rules
15000021283	6.980902857	43.3137085	0.068026417	ItI
4710091110491	6.966410256	43.3137085	0.069060631	UtU
44738072342	6.920714286	43.55634919	0.075127867	UtU
4713045026183	6.907111111	43.55634919	0.076106404	UtU
4710734001155	6.905965517	43.55634919	0.076188893	UtU
4710020150017	7.411813953	47.79898987	0.087392462	UtU
4710177012060	6.825666667	44.97056275	0.097976743	UtU
4710088414687	7.078878437	50.14213562	0.134152687	UtU

Chapter 6

Conclusions and Future Work

This thesis details the research made about recommender systems and the development of a system that can recommend products taking into consideration the target user's indoor location.

Was initially intended to respond to three types of scenarios:

- When the customer is planning the shopping trip and the shopping list is being created
- When the customer is actually shopping in store with navigation being used
- When the customer is actually shopping in store with navigation not being used

When the project ended it was safe to say that the system, when correctly configured, can give an answer to these scenarios. Recommendations are an important part of many stores, with recommendation systems being incorporated in almost all online stores. Recommender systems can help customers not only by discovering new products that he could like, but also, reducing the time's spent on searching for new experiences. The system developed seems to provide meaningful recommendations while reducing, if necessary, the time that the customer would spent on a store. It was rewarding to see, that sometimes by changing the user's path in some cases by zero weight, the customer would pass by shelf's with products that he could like.

6.1 Future Work

There are great opportunities in recommendation system in my opinion. While developing this thesis I found some work that could be developed:

- Find the best parameters value for the recommendations system based on simple keywords, describing the user context, for example "OutOfTime" if the user does not have a lot of time for the shopping trip. While the system developed can be configurable, it is unknown the best inputs for each scenario.

- Better comparison between association rules and recommendations generated by CF algorithms
- Testing different metrics for the user-item matrix
- Record rejected recommendations and give low affinity on user-item matrix when generating a new model

References

[AMRT11] Gediminas Adomavicius, Bamshad Mobasher, Francesco Ricci, and Alexander Tuzhilin. Context-Aware Recommender Systems. *Journal of Software*, 23(1):1–20, 2011.

[Ana16] Sousa Ana. *Market-based Higher Education Program Recommendation*. PhD thesis, Faculdade de Engenharia da Universidade do Porto, 2016.

[AT10] Gediminas Adomavicius and Alexander Tuzhilin. *Recommender Systems Handbook: A Complete Guide for Research Scientists and Practitioners*. Springer, 2010.

[Bur07] Robin Burke. Hybrid web recommender systems. *The adaptive web*, pages 377–408, 2007.

[BV09] Toine Bogers and Antal Van Den Bosch. *Collaborative and content-based filtering for item recommendation on social bookmarking websites*, volume 532. 2009.

[CLA+03] Dan Cosley, Shyong K. Lam, Istvan Albert, Joseph a. Konstan, and John Riedl. Is Seeing Believing? How Recommender System Interfaces Affect Users' Opinions. In *Proceedings of the Conference on Human Factors in Computing Systems (CHI '03)*, pages 585–592, 2003.

[Cor13] Brucato Corinne. *the Traveling Salesman Problem by Corinne Brucato B . A ., Sonoma State University , 2010 M . S ., University of Pittsburgh , 2013 Submitted to the Graduate Faculty of the Department of Mathematics in partial fulfillment of the requirements for the degree*. PhD thesis, University of Pittsburgh, 2013.

[DM11] Gilda Moradi Dakhel and Mehregan Mahdavi. A new collaborative filtering algorithm using K-means clustering and neighbors' voting. *2011 11th International Conference on Hybrid Intelligent Systems (HIS)*, pages 179–184, 2011.

[HK15] Htet Htet; Hlaing and Kyi Thar Ko. Location-Based Recommender System for Mobile Devices on University Campus. In *Proceedings of 2015 International Conference on Future Computational Technologies*, pages 204–210, 2015.

[KBV09] Y. Koren, R. Bell, and C. Volinsky. Matrix Factorization Techniques for Recommender Systems. *Computer*, 42(8):42–49, 2009.

[Lin00] Weiyang Lin. *Association Rule Mining for Collaborative Recommender Systems*. PhD thesis, 2000.

[Liu14] Xiaohu Liu. *Context-Aware Recommender Systems for Implicit Data*. PhD thesis, 2014.

REFERENCES

[LSEM12] Justin J. Levandoski, Mohamed Sarwat, Ahmed Eldawy, and Mohamed F. Mokbel. LARS: A location-aware recommender system. In *Proceedings - International Conference on Data Engineering*, volume 1, pages 450–461, 2012.

[NC] Igor Naverniouk and Frank Chu. The shortest path problem.

[Ote15] Abayomi Moradeyo Otebolaku. *Context-Aware Personalization for Mobile Multimedia*. PhD thesis, Faculty of Engineering University of Porto, 2015.

[PG13a] Nikolaos Polatidis and Christos K. Georgiadis. Mobile recommender systems: An overview of technologies and challenges. *2013 2nd International Conference on Informatics and Applications, ICIA 2013*, pages 282–287, 2013.

[PG13b] Nikolaos Polatidis and Christos K. Georgiadis. Recommender Systems: The Importance of Personalization in E-Business Environments. *International Journal of E-Entrepreneurship and Innovation*, 4(4):32–46, 2013.

[RRSK11] Francesco Ricci, Lior Rokach, Bracha Shapira, and Paul B. Kantor. *Recommender Systems Handbook*, volume 53. 2011.

[SKKR01] Badrul Sarwar, George Karypis, Joseph Konstan, and J. Riedl. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th . . .* , volume 1, pages 285–295, 2001.

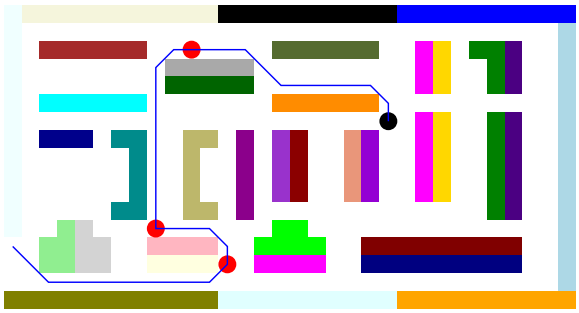
[SMU14] Degli Studi, D I Milano, and Bicocca Universit. *Advancing Recommender Systems from the Algorithm , Interface and Methodological Perspective*. PhD thesis, 2014.

[UF98] Lyle H. Ungar and Dean P. Foster. Clustering methods for collaborative filtering. *AAAI Workshop on Recommendation Systems*, pages 114–129, 1998.

[YSC+13] Hongzhi Yin, Yizhou Sun, Bin Cui, Zhiting Hu, and Ling Chen. LCARS: a location-content-aware recommender system. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 221–229, 2013.

Appendix A

Appendix



Graph Points

- Point nothing
- Select Position
- Select Shoping Points

User ID

1001679

Your Basket

×20515447 ×10742202211 ×4103040328120

Path Weight

43.31370849898476

Recommender config

Rec Conf

0.1

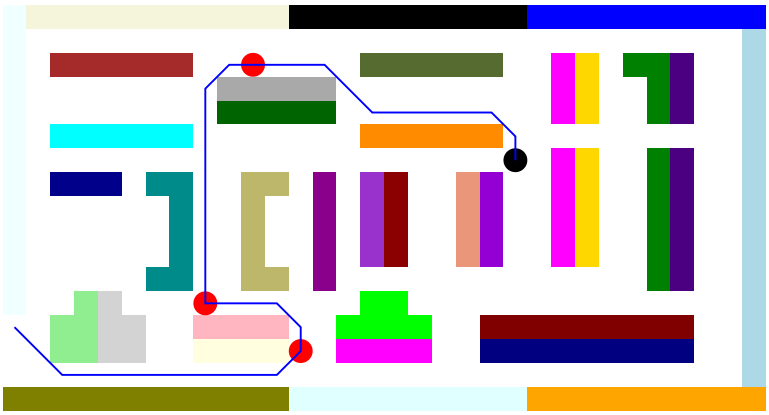
Rec Path

0.9

Recommendation List

ProductId	Score	Path Weight	Sort Value	From	View Path	Add Basket
20515423	0.4889	43.31370849898477	0.06865471153200364	Rules	View	Add
20515430	0.4667	43.31370849898477	0.07272107452103377	Rules	View	Add
15000021283	6.980902857142858	43.31370849898477	0.013605283374109186	Itl	View	Add
4710091110491	6.966410256410258	43.31370849898477	0.0138121262999196	UtU	View	Add
44738072342	6.920714285714288	43.55634918610404	0.019494605735994958	UtU	View	Add
4713045026183	6.907111111111112	43.55634918610404	0.019690313091264143	UtU	View	Add
4710734001155	6.9059655172413805	43.55634918610404	0.019706810930169006	UtU	View	Add
4710177012060	6.8256666666666685	44.970562748477136	0.04962297149949765	UtU	View	Add
4710020150017	7.411813953488375	47.79898987322333	0.09624303703159609	UtU	View	Add
4710088414687	7.078878436676392	50.14213562373095	0.14373583637941623	UtU	View	Add

Figure A.1: Results 1



Graph Points

- Point nothing
- Select Position
- Select Shoping Points

User ID

1001679

Your Basket

×20515447 ×1074202211 ×4103040328120

Path Weight

43.31370849898476

Recommender config

Rec Conf

0.9

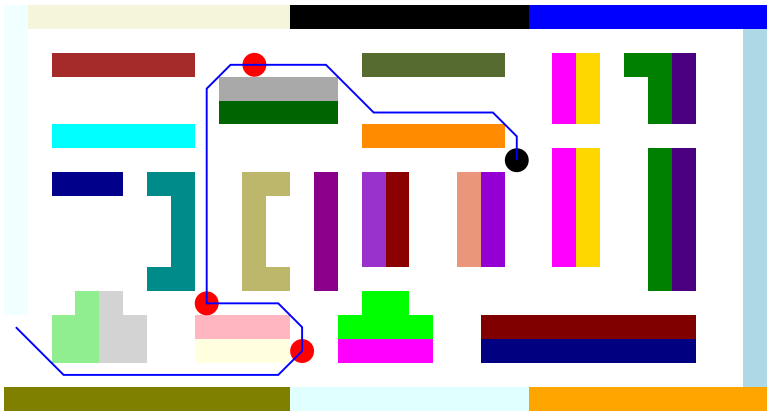
Rec Path

0.1

Recommendation List

ProductId	Score	Path Weight	Sort Value	From	View Path	Add Basket
20515423	0.4889	43.31370849898477	0.6178924037880315	Rules	View	Add
20515430	0.4667	43.31370849898477	0.6544896706893025	Rules	View	Add
4710020150017	7.411813953488375	47.79898987322333	0.07854188682146804	UtU	View	Add
15000021283	6.980902857142858	43.31370849898477	0.12244755036698135	Itl	View	Add
4710091110491	6.966410256410258	43.31370849898477	0.1243091366992751	UtU	View	Add
4710088414687	7.078878436676392	50.14213562373095	0.12456953750779189	UtU	View	Add
44738072342	6.920714285714288	43.55634918610404	0.1307611279685701	UtU	View	Add
4713045026183	6.907111111111112	43.55634918610404	0.13252249416599274	UtU	View	Add
4710734001155	6.9059655172413805	43.55634918610404	0.13267097471613654	UtU	View	Add
4710177012060	6.8256666666666685	44.970562748477136	0.14633051515610582	UtU	View	Add

Figure A.2: Results 2



Graph Points

- Point nothing
- Select Position
- Select Shoping Points

User ID

1001679

Your Basket

×20515447 ×10742202211 ×4103040328120

Path Weight

43.31370849898476

Recommender config

Rec Conf

0.5

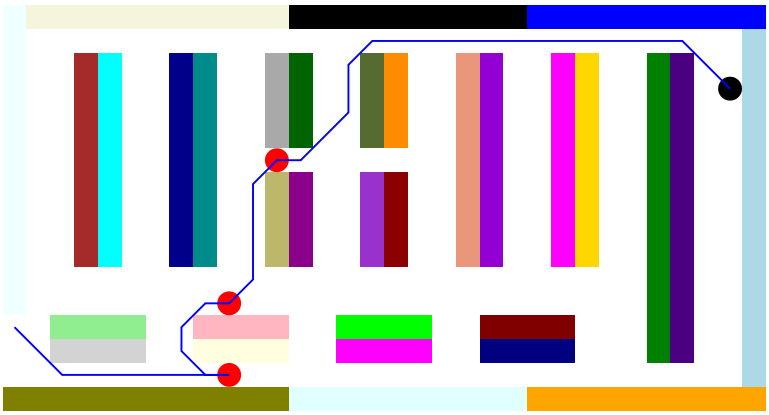
Rec Path

0.5

Recommendation List

ProductId	Score	Path Weight	Sort Value	From	View Path	Add Basket
20515423	0.4889	43.31370849898477	0.3432735576600175	Rules	View	Add
20515430	0.4667	43.31370849898477	0.3636053726051681	Rules	View	Add
15000021283	6.980902857142858	43.31370849898477	0.06802641687054527	Itl	View	Add
4710091110491	6.966410256410258	43.31370849898477	0.06906063149959735	UtU	View	Add
44738072342	6.920714285714288	43.55634918610404	0.07512786685228252	UtU	View	Add
4713045026183	6.907111111111112	43.55634918610404	0.07610640362862844	UtU	View	Add
4710734001155	6.9059655172413805	43.55634918610404	0.07618889282315276	UtU	View	Add
4710020150017	7.411813953488375	47.79898987322333	0.08739246192653206	UtU	View	Add
4710177012060	6.8256666666666685	44.970562748477136	0.09797674332780174	UtU	View	Add
4710088414687	7.078878436676392	50.14213562373095	0.13415268694360405	UtU	View	Add

Figure A.3: Results 3



Graph Points

- Point nothing
- Select Position
- Select Shoping Points

User ID

1001679

Your Basket

×20515447 ×10742202211 ×4103040328120

Path Weight

45.556349186104036

Rec Conf

0.1

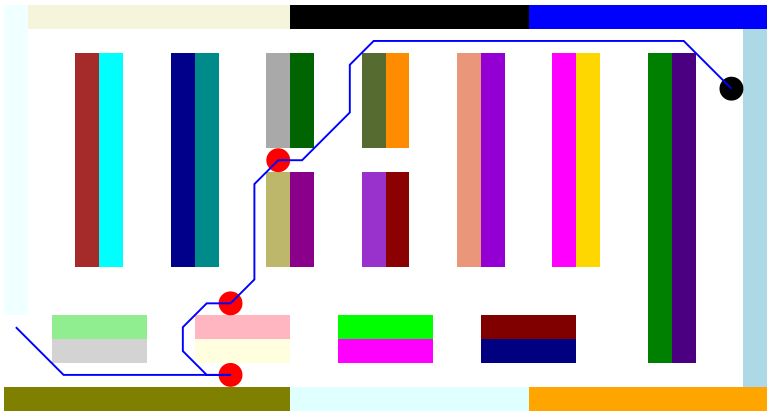
Rec Path

0.9

Recommendation List

ProductId	Score	Path Weight	Sort Value	From	View Path	Add Basket
20515423	0.4889	45.55634918610404	0.06865471153200363	Rules	View	Add
20515430	0.4667	47.55634918610404	0.1113838999243894	Rules	View	Add
4710091110491	6.966410256410258	45.55634918610405	0.013812126299919734	UtU	View	Add
4710734001155	6.9059655172413805	45.55634918610405	0.014679149518938527	UtU	View	Add
4710177012060	6.8256666666666685	45.55634918610405	0.015841895811318465	UtU	View	Add
44738072342	6.920714285714288	46.384776310850235	0.030685680135134015	UtU	View	Add
4712425010392	6.896952380952382	46.38477631085024	0.031027773133440177	UtU	View	Add
4710020150017	7.411813953488375	47.55634918610404	0.046295749707876005	UtU	View	Add
15000021283	6.980902857142858	47.55634918610405	0.052268108777464975	Itl	View	Add
4710088414687	7.078878436676392	49.21320343559643	0.08167362406315755	UtU	View	Add
8801037010497	6.923172413793105	53.55634918610405	0.1597207705920509	UtU	View	Add

Figure A.4: Results 4



Graph Points

- Point nothing
- Select Position
- Select Shoping Points

User ID

1001679

Your Basket

×20515447 ×10742202211 ×4103040328120

Path Weight

45.556349186104036

Rec Conf

0.9

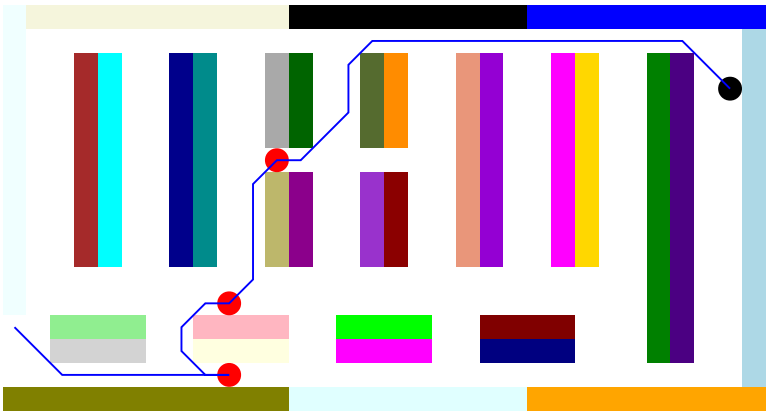
Rec Path

0.1

Recommendation List

ProductId	Score	Path Weight	Sort Value	From	View Path	Add Basket
20515423	0.4889	45.55634918610404	0.6178924037880315	Rules	View	Add
20515430	0.4667	47.55634918610404	0.6587855401785643	Rules	View	Add
4710020150017	7.411813953488375	47.55634918610404	0.07299218822994358	UtU	View	Add
4710088414687	7.078878436676392	49.21320343559643	0.1176737361393187	UtU	View	Add
4710091110491	6.966410256410258	45.55634918610405	0.12430913669927511	UtU	View	Add
15000021283	6.980902857142858	47.55634918610405	0.1267434198562431	Itl	View	Add
44738072342	6.920714285714288	46.384776310850235	0.13200458067958554	UtU	View	Add
4710734001155	6.9059655172413805	45.55634918610405	0.13211234567044425	UtU	View	Add
4712425010392	6.896952380952382	46.38477631085024	0.13508341766433973	UtU	View	Add
4710177012060	6.8256666666666685	45.55634918610405	0.14257706230186368	UtU	View	Add
8801037010497	6.923172413793105	53.55634918610405	0.14602779737512864	UtU	View	Add

Figure A.5: Results 5



Graph Points

- Point nothing
- Select Position
- Select Shoping Points

User ID

1001679

Your Basket

×20515447 ×10742202211 ×4103040328120

Path Weight

45.556349186104036

Rec Conf

0.5

Rec Path

0.5

Recommendation List

ProductId	Score	Path Weight	Sort Value	From	View Path	Add Basket
20515423	0.4889	45.55634918610404	0.3432735576600175	Rules	View	Add
20515430	0.4667	47.55634918610404	0.38508472005147687	Rules	View	Add
4710020150017	7.411813953488375	47.55634918610404	0.05964396896890979	UtU	View	Add
4710091110491	6.966410256410258	45.55634918610405	0.06906063149959742	UtU	View	Add
4710734001155	6.9059655172413805	45.55634918610405	0.07339574759469138	UtU	View	Add
4710177012060	6.8256666666666685	45.55634918610405	0.07920947905659106	UtU	View	Add
44738072342	6.920714285714288	46.384776310850235	0.08134513040735977	UtU	View	Add
4712425010392	6.896952380952382	46.38477631085024	0.08305559539888996	UtU	View	Add
15000021283	6.980902857142858	47.55634918610405	0.08950576431685403	Itl	View	Add
4710088414687	7.078878436676392	49.21320343559643	0.09967368010123812	UtU	View	Add
8801037010497	6.923172413793105	53.55634918610405	0.15287428398358976	UtU	View	Add

Figure A.6: Results 6