Master in Artificial Intelligence

Master Thesis Report

Explainability on Recommender Systems using Contextual Data

Author: Asier Gutiérrez Fandiño

Supervisor: Maria Salamó Llorente
Departamento de Matemáticas e Informática (UB)

Co-Supervisor: Ludovico Boratto
Eurecat - Centro Tecnológico de Catalunya

April 14, 2020

FACULTAT D'INFORMÀTICA DE BARCELONA (FIB)

FACULTAT DE MATEMÀTIQUES (UB)

ESCOLA TÈCNICA SUPERIOR D'ENGINYERIA (URV)

UNIVERSITAT POLITÈCNICA DE CATALUNYA BARCELONATECH

UNIVERSITAT DE BARCELONA

UNIVERSITAT ROVIRA i VIRGILI

Acknowledgements

To Maria and Ludovico because of their necessary support, priceless knowledge and infinite patience.

To my family for such an economic effort.

Abstract

Recommender Systems (RS) are a fundamental part of any relevant e-commerce website, and the inclusion, improvement and optimization of this kind of systems is a growing trend. As these systems evolve, people are more demanding with the accuracy and explainability of their recommendations.

The goal of this work is to create a RS that provides simple explanations of the recommendations that it provides. These explanations are obtained by finding the different relations that can be interpreted from the data of any dataset.

This work shows that recommendations can be explained in an easy way with Machine Learning (ML) algorithms that let contexts be understood from different perspectives. In this master thesis, every ML algorithm provides recommendations. Since algorithms can be explained, recommendations provided by them can also be explained in natural language, and, in some cases, the explanations are even provided with a helper picture.

ML algorithms used in this master thesis provide recommendations with some features that can be queried by the user in order to meet user's current recommendation expectations or filtering settings. Users, therefore, obtain not only explainability, but a very open recommender system where dark patterns do not exist and there are neither boring nor repetitive recommendations. Users can easily explore explainable recommendation spaces that are prepared specifically for each one of them, with parameters that help them decide implicitly the ratio of exploration and exploitation.

All this work is presented in a framework that allows to encode as much context information as wanted by capturing as best recommendation space as possible at the time explainability results evolve even richer.

This framework comprehends all the different representation, aggregation, computing and visualization techniques coded in a modular and extendable way so that it can be applied to any recommendation scenario.

A variety of datasets have been tested in many ways showing surprisingly positive results of recommendation explanation capabilities in all the datasets.

Contents

Chapter 1

Introduction

Recommender Systems are a large Artificial Intelligence field where Machine Learning systems and Deep Learning systems are applied. The goal of a RS is to provide suggestions of products/services/actions to users/agents that are likely to be interested in. Recommendation task is not an easy task as it hides different sub-problems. All of these sub-problems can also be seen from the point of view of Psychology, Business and Marketing among others: what is a good recommendation? How is a recommendation interpreted keeping both businesses' and users' privacy concerns? How can data be encoded from different sources to provide a recommendation? How can a rating be better predicted given a user-product pair? How to allow the user parameterization or input to make the system provide better recommendations?

The goal of this master thesis is to research on context-aware Recommender Systems [1, 2] with explainability, and provide a RS that uses context to obtain explanations. This context encoding is simple, customizable and expandable so that many contexts can be encoded on demand, without adding complexity to the system. The goal is not to make better regressions of the ratings, but to provide explanations that can be understood by both users and platform owners, thanks to Machine Learning algorithms that are traceable. With the system of this master thesis, users can explore the recommendation space autonomously with intuitive parameterization, obtaining explanations of their recommendations.

The core part of a Recommender System is usually the rating regression. The regression of a rating is a prediction of what the RS thinks the user will provide as a rating. This regression is provided often by an Artificial Neural Network acting as a regressor that has as input, typically, data from both item and user and outputs the expected rating. Some systems perform this task implicitly and others explicitly. In this proposed system the aim is to provide a Recommender System that can deliver at least the same results as other simple but yet competitive systems.

A production-ready Recommender System should always try to balance between exploration and exploitation, which means that the user should receive recommendations that will like for sure, combined with other recommendations that will make

the user discover new products whilst the system is understanding new compatibilities and behaviors. The aforementioned are relationships that the user will validate or not with the feedback system. The goal in this master thesis is to identify the most basic features that comprise exploration and exploitation, provide them to the user in a safe way for the Recommender System and measure the impact on the different metrics that will be used.

Explainability [3, 4, 5, 6] is a topic that is gaining relevance in AI. Indeed, it is necessary in Recommender Systems because there is usually a large amount of products to be recommended, so that selecting the one with the highest expected rating is not the best product choice. Dark patterns are considered unethical and users are tricked in one way or another to buy certain things. There should be an explanation that will help the user understand why the system is working in some way and how to change or maintain that behavior via feedback or new purchases. The use of Artificial Neural Networks (ANN) in Recommender Systems [7, 8] yields positive results but low explainability. In this master thesis, the approach is to use ANNs as helper but the core will be the context explanatory techniques providing simple explanations. Thus, the user will be able to modify recommendation parameters according to the moment, check the explanations and iteratively modify them to get better recommendations.

In order to understand the reasoning behind this work and the relevance of it, other works are analyzed and discussed in Chapter 2. The factors to take into account in this Chapter are not only results, but also the previously mentioned open sub-problems in Recommender Systems. In this Chapter 2, a series of ideas on how to solve these problems will be exposed, whereas the proposal Chapter will explain why the selected one is an appropriate one and why the others did not work or have not even been developed/tested.

In Chapter 3, the Proposal describes all the scientific background behind the solution and how this scientific background is interpreted for this problem; the algorithms used are also explained along with the modifications made for making them recommend. A framework of these characteristics (easy-start with any dataset, online-ready, extendable, etc.) needs an architecture that takes into account all the possible inclusions of code, datasets and changes; in other words, the architecture has to be as much generic as possible at the time, so as to be fully adaptable to architectural changes with little effort. In this Chapter, the architectural solution with its features and limitations will be proposed. It is understandable that this framework will not be exclusively used with the datasets it has been tested with; that is why some extra patterns, scalability issues and advices are also provided. The Chapter of that contains future work will be pointing in the same direction.

In order to check whether this framework is providing competitive results, the framework contains a complete evaluation step with many tests. Chapter 4 is for the evaluation and includes an explanation of how the training-testing split is conducted, as well as the methods used to compare and the reason for the selected metrics. Firstly, the methodology for the testing is defined and justified. Then, datasets

used for testing are analyzed and their different features are mentioned. After that, the whole setup is described.

Finally, in the Evaluation Chapter, the results from the different tests are shown and discussed. The most outstanding results are focused on the following issues: how regression approximates to standard recommendation methods, why the alternation of the recommendation selection method brings better results, how different datasets affect the explanatory capabilities of the system and how results, computations and capabilities could be enhanced.

In Chapter 5 the conclusions combine the initial thoughts with the final thoughts, answers important questions and provides first insights for the further work. The Future Work Section present in that Chapter describes how to improve the current system in terms of the regressor and the recommender itself.

Chapter 2

Related Work

This part starts describing a wide view of RS [9] explaining which is the current state of the art and the problems derived. Then, the related work focuses on explainability [4, 5, 6] with Language Models (LM) and Rule Mining; a complete review is given for works regarding graphs and embeddings [10, 11, 12, 13] as they offer competitive ways to encode context [10]. Artificial Neural Network architectures are reviewed [7, 8] due to the fact that they offer state-of-the-art results not only in Recommender Systems but in most of the Artificial Intelligence tasks (general NLP [14], Question Answering [15], Chatbots [16], Image Classification [17], Image Retrieval [18], Human Pose Estimation [19], Graph Embedding [20], Graph Alignment [21], Reinforcement Learning [22], etcetera). In this master thesis one of the goals is to satisfy user requirements explicitly and this can be done via constraints or using efficient searches, both contingent upon user interaction. For this purpose, literature regarding constraint implementation in RS [23, 24] and encoding and optimization of recommendations derived from user interaction [25, 26, 27] are analyzed.

The work of Ferrari et al. [9], which analyzes the current status of Recommender System research is fully inspirational for this work. This paper points out that all Recommender Systems proposed in top research conferences are difficult to reproduce, if not impossible, and systems are not providing positive results at all. The paper states that complex algorithms are often outperformed by heuristics. The positive aspect is that a heuristic is explainable while a complex algorithm is not. Accordingly, the proposed solution uses simple algorithms that will help both scientists and users understand what is going on with recommendations. Scientists can debug the algorithms proposed easily and users will be provided with an explanation on how the system has computed the recommendation.

Ferrari et al. warn that the use of incorrect baselines that are not suitable for a dataset or are not correctly fine-tuned prevent progress from happening. This makes a proposed method apparently be a research progress but actually it is not. In this proposed solution, the regressor is evaluated against standard methods and positive results are obtained. Neither the regressor nor the baselines are tuned and both provide almost the same results; sometimes the winner is the proposed regressor and the rest of the times the baselines. Since this master thesis does not aim at obtaining the best regression, the results are positive enough to continue using it.

The above-mentioned work only considers the top-N recommender systems and deep-learning-based methods that could be reproduced. In this master thesis, top-N recommender systems are not used as baselines because the evaluation metrics are different (precision, recall, MAP) from the ones used when regressing the expected rating (MAE, RMSE and FCP). It could also include the evaluation metrics of the mentioned work; however the goal of this Recommender System is not to provide the exact recommendations of the product that the user would buy by hiding some products in the train-test split but let the user interact with some parametric variables to personalize its recommendation. The reason for this is the idea that a user may have been previously biased by many factors: the website, the recommender system or factors involving the day in which the rating was provided, among others.

Lu et al. [4] claim that they use a Language Model (LM) to generate explanations of their model; nonetheless, they do not show how the Language Model is generated, they do not present an example explanation and they only state that the evaluation of the explanations is attained through perplexity. Apparently, those explanations can be used for any model. Similar to the proposal of this master thesis, they use embeddings to represent items and users with encoder-decoder architecture, but they use matrix factorization to predict the expected rating. This leads to one sole way of obtaining recommendations, which means that there is only one dimension of exploration.

Tsukuda et al. [5] utilize rules that are formed by predicate logic to explain recommendations. Rules have to be hand-crafted and, thus they are costly to implement. Predicate logic rules do not cover all the recommendations possible for the user; hence there is a limitation in that sense. Since there is not a possible user input, the recommendations cannot be driven towards certain explorations of the user recommendations space. This explainability is limited only to rules and they do not consider other types of recommendations; users do not understand predicate logic and the explanations are considered to be too complex.

Wainakh et al. [6] propose the use of Privacy Preserving Association Rule Mining (PPARM) technique, instead of Apriori association rules, in order to efficiently generate explainable recommendations based on association rules. The technique is not as explainable as Apriori and, a minimum confidence is needed for the associations to exist. The Recommender System developed in this master thesis executes a filtering of items that do not fulfil minimum requirements and Apriori algorithm sets the minimum confidence, so that it captures global rules without affecting the privacy or leaking a privacy exploitation vector, because it is showing rules that are very generic and no data about users are provided. It is important to keep in mind that, when a very low minimum confidence is established, even an efficient Apriori algorithm can take too long to compute all the rules.

Sun et al. [10] propose recurrent embeddings for dealing with different contexts in graph form. These contexts, along with users and items, are represented in form of a multiplex network. It is one of the first issues that was considered for this thesis, since it could improve the results of the regressor. For a production system with

many contexts represented in graph data, if there is enough data, it is recommended to produce embeddings of the whole multiplex network. Adding the contextual data could contribute little but still relevant information to capture all the variability. From the point of view of software design, there are more challenges because the system proposed in this thesis only needs a Dataset Adapter and a configuration file. From the point of view of interpretability, the embeddings computed in the work by Sun et al., knowing the node class and computing the distance is not as trivial as in the proposed solution. When working with embeddings, having at least the availability to know how nodes are related is important when the system does not have explainability; this one does not have explainability in any case and has difficult comparison and interpretation of node embeddings.

Palumbo et al. [11] suggest graph representation and top-N recommendation without noticing that a matrix can be represented as a graph and matrix factorization does the same. They obtain better results because the representation in graph form and the technique they used considers the recursion of the graph. In other words, the graph is fully traversed while some matrix factorization techniques lack in that sense. Embeddings of this thesis cannot only provide good regressions, but they can also be compared with cosine distance; plus there is an explanation of the recommendation. Even if they are only using graphs to provide recommendations, they do not show the explanation of the path that the recommender has followed in order to provide an explainable recommendation.

Lyu et al. [12], instead of obtaining an embedding from directly looking up by node ID, calculate an embedding from composition of nodes in order to reduce the noise. This is somehow similar to making communities of nodes and then obtaining the community ID of that node instead of the embedding. This would not work when two embeddings are being used to compute the regression as the information would be too generic to make a precise prediction. That is why nodes in this master thesis are aptly separated and compute its embeddings separately. As the dataset grows and new context embeddings (for example, friend embeddings) are included, the proposed generalization by Lyu et al. can make the regressor lose information for the prediction. The work of this thesis lets make a composition of different embedding methods in order to be less restricted by the particular embedding algorithm assumptions and/or reduce the noise of embeddings.

Rashed et al. [13] propose to compute embeddings from user and items and add relevant information to them such as age and gender for the users and release date and genre for items (in the movies domain). This technique is fast because it works with matrix factorization, can produce positive results and takes into account a typical Recommender System deployment scenario, which is a database where there is more information of users and items than only their ID. This system is less explainable than a matrix factorization technique because the process of generating the recommendation takes an embedding step, an adaptation to the matrix to be square matrix and a normalization of scores.

Christakopoulou et al. [7] propose an adversarial attack on Recommender Systems based on Generative Adversarial Networks (GAN) [28]. The proposed GAN creates fake user profiles that interact with the system and try to modify recommendation structures in order to make real users get inadequate recommendations. It is relevant to analyze if it would have an impact on this system; it would not actually have an impact on this system because there is a large number of contextual recommenders involved in a user recommendation. The greater impact that the recommender system would receive is placed in the regression step where the numbers may vary a little bit. But since the embeddings would not be modified for each data inclusion and the embeddings are more generic, the impact would still be minimum. As for the recommendation, this contextual recommender is not based on top-N recommendations and, thus, the impact on the regressor would not be reflected in the recommendations at all. The most affected recommendations would occur if a user would demand highly rare recommendations. In that case, an attack could happen; yet, since there are a lot of contexts involving the item, it would have to be a very comprehensive attack. That attack in any production system with many contexts should be identified as an attack. For instance, if the context is being exploited by adding a considerable number of friends to a network, of products being weirdly voted, of user being created, of communities being changed, the production systems monitoring should notice that at a glance.

Liu et al. [8] propose a Deep Generative method in order to consider implicit interactions. This model yields positive results, with no explainability, though. Another objection is that the method considers any kind of interaction as feedback when not all the interactions in items from the user might be positive feedback. A click, a "like" and a purchase are interpreted as interactions, thus forgetting that sometimes a click can be done accidentally or by confusion. The model proposed in this master thesis, since it is modular, could include all the ways of interactions separately in an easy way and could also explain the recommendations in those contexts. It would also let the user decide the importance that it gives to that kind of feedback via input. On the contrary, the model of Liu et al. could not model any other feature. As it is based on error minimization, the model will try always to minimize the error leading to the exploration versus exploitation disjunctive, and choosing exploitation in the form of optimization.

Bonner et al. [23] seek to maximize the outcome at the time they are providing positive recommendations for the user, but this might not be considered fair. The work presented in this thesis, as it stores potential recommendations for the user and lets the user demand its current exploration preferences, the Recommender System can omit the ones with low outcome or even reorder them by applying some scoring. This approach could be attainable, yet it is against the nature and the goal of this Recommender System, that is, to avoid hidden dark patterns and let the user explore whilst it exploits the system.

Makhijani et al. [24] developed an offer recommendation online engine for large-scale dataset in order to provide users with the best offers according to constraints. The constraints that the system concentrates on are eligibility (if the offer fits the

user, the user will use that offer and therefore a purchase is done) and capacity (offers provided will maximize the income and will not produce losses; in other words, it is not wasting the company's money). They generate a min-cost flow network at the time they maximize the compatibility to the user. This is done by training a policy. The work of this master thesis can also be used for providing offers; with the recommendations on the database, it can maximize the income and minimize the cost, first setting the scores of the products/offers on the database and then querying a sorting for the specific user. Its best characteristic is that it can provide rich and specific offers at the time they are explainable.

Al-Ghossein et al. [25] suggest a hybrid Recommender System based on Adaptative Window based Latent Dirichlet Allocation (AWILDA) to solve the cold start problem when adding the online inclusion of new products. This model considers clicks as interactions, but as mentioned above, a click can be made by mistake. Accordingly, this could be a weak model, sensitive to attacks. The proposed solution in this master thesis does not implement in the code the solution of the cold start problem, since, depending on the domain, the implementation varies and it is up to the developers; at least, some relevant guidelines are provided in order to maintain the same explainability and accuracy.

Harambam et al. [26] created an apparently User Experience-centric (UX-centric) Recommender System where users are told to select the value of three preferences. Users are also told to provide five values for recommendation output ordering that include the same preferences plus two new ones related to similarity and dissimilarity. Then, users are told to select which recommendation algorithm should be used: Collaborative Filtering, Content Based, randomly, by notion of serendipity (they do not explain how) or ideologically (they only explain how they order them but not how they detect automatically the ideology). All these types of user inputs, orderings and recommendation engine, apart from generating a combinatorial explosion difficult to test, can confuse the user. The solution proposed in this master thesis uses only five value inputs from 0 to 1 and let the user shorten the recommendations by maximising the expected rating or by maximising the explainability. It is a much simpler way, owing to the fact that the user is not aware of how recommendations are generated internally, unless the recommendation fits its actual and current needs.

Winecoff et al. [27] perform an interesting analysis in psychologically informed similarity function by making Amazon Mechanical Turk users select between two dress alternatives for a target dress. With 13,452 observations they train the Tversky model to learn the similarity function with the assumptions they made. Although this research has been conducted meticulously, the variety of dresses and observations might not be sufficient to obtain a fully generic trained similarity function. The feature space of dresses is large for all the dresses and thus it is a very restricted similarity function. It is a highly useful study for a small shop with very limited number of products over the time or items that share a certain style. The solution of this work seeks similarity in terms of communities of linked products and patterns in data, driven by general rules and products that share a common behavior, all this in an explainable way. In contrast, the work by Winecoff et al., once the parameters

are learnt, cannot express why two items share similarities and, consequently, they cannot explain the recommendation provided. Another objection is that, in case of massive overfitting, it can be providing recommendations of nearly the same products. And regarding the clothing domain, dressing extremely similar clothes might be awkward.

The works discussed above follow two main research lines that are improving the alignment on user preferences and developing explainable models; some of the works take into account implicit user interaction to help capture user needs. There is a large room for improvement in explainability because nobody has thought that basic Machine Learning algorithms might add simple explainability and that all the ML algorithms can be aggregated (for the mentioned purpose). Aligning this explainability techniques with explicit user interaction can improve both explainability and user perception of the RS implicitly. This will lead to users finding what items get users preferences satisfied without the need for the system to decide between exploration and exploitation.

Chapter 3

Proposal

3.1 Context Techniques

As in Complex Networks detecting what is a good Community Detection or in Machine Learning what is a good clustering, a good context understanding is a difficult task. On the basis that a recommendation environment considers or stores much different contextual information[1], the problem becomes even more complex and harder to tackle with a 'master' algorithm. Firstly, this is because that perfect algorithm still does not exist and, secondly, because all the algorithms make assumptions that may not fit in a certain context.

Elaborating an algorithm that will consider all assumptions is also impossible, because it is compulsory to relax some constraints in order to exploit benefits from data. Existing algorithms produce positive results and understand the context in some way or another, trying to optimize certain metrics or relationships.

A context can also be represented in many different ways. A graph representing a context can be of different types: directed, undirected, with nodes of only one kind (as a complex network), with nodes of different kinds with many interactions, etcetera. Points in space can also represent contextual information, but how the dimensions are scaled and the points are distributed depends on the representation. Context can also be represented as a row in a matrix where order can matter or not.

The proposed solution to this problem is to make as much representations of the contexts as wanted and run algorithms that will help make different understandings of the contexts. In this way, understandings will capture as much information as possible from the contexts.

Understanding the context is crucial to provide a recommendation because it can place a user optimally in the recommendation space. For instance, understanding the connections of friends of a certain user is relevant to know which are the influences

[1]In Recommender Systems, contextual information is additional data representing useful insights of the environment such as friends, tags, product features, user features, timestamps and so on. In another example of a more specific scenario, that could be one related to movies, contextual information can be formed by movie actors, directors, countries, genres and locations.

that the user is being exerted. However, this contextual understanding of influence is not unique, it can be understood in many different ways.

As stated in the first paragraph, defining what is a good group, community or cluster is difficult not only for algorithms, but also for people. Continuing the example of a friends' network of a certain system where a recommender system could be applied, how to understand the flow of influence or even select correct group of friends is complicated. In a gaming scenario of recommending games (for instance, Steam), best friends may not be the ones playing together most of the time. Probably, though, greater influence can appear in that situation for certain cases. For example, purchasing a game related to war could be an influence from the friend that is playing with the target player that type of games for a long time, but it could not be the case of games related to MMORPG games.

There are infinite reasons to buy any product, and computing them is not feasible. Instead, executing or exploiting the most meaningful context understandings will provide a strong approximation of the most suited recommendations. Still, there will be some recommendations that will not be covered because the influences, needs and reasons are out of the scope of information of the system. A user can buy a product because its favourite YouTube star has bought it, because a princess has worn that dress, because a friend that is or is not a user of the system has recommended it, because it has been seen in an advertisement on television, because the product appeared in an old film and looking on the internet for that product has attracted the user to buy a new version of it, and so on.

Adding the results of very basic and traceable algorithms can be helpful to understand the context because, as it can be noticed from previous paragraphs, there are infinite reasons, but all these reasons are interconnected and explainable.

The Maximum Spanning Tree (MST) [29] algorithm is one of the selected algorithms for this work. Given a graph that can be weighted or not, the algorithm constructs a tree in which nodes with higher interactions appear at the highest levels of the tree and nodes with smaller interactions appear at the lowest levels of the tree. Nodes cannot be repeated and the tree is not recursive, which is one of the assumptions of the algorithm where there could be some lack of information.

MST captures a lot of information from a graph. This computation, since it sorts the nodes, is useful to know the relevance of a node. This tree can also be understood as the result of hierarchical clustering [30]; the same cited authors, in the paper, define the resulting tree as a transport backbone of the network. Nodes that are at higher levels are more connected. When traversing the graph from one node to another, these nodes with more connections are more likely to be used, because their links are stronger. Nodes in higher levels are like key nodes of the graph, especially in Scale-Free networks. Nodes in higher levels are also more general, and nodes in lower levels tend to be more specific.

The intuition of MST translated to recommender systems is the same, but the interpretations will differ in the context where they will be encoded. In the context of item-item relationship, where the weight is the number of times that both items are bought by a user, the interpretations that can be done are the same as the ones mentioned in the previous paragraph. Items that are at the top of the tree are very generic items that everybody buys, whilst the products that appear at the leaves are very specific. Items can be ranked by their depth in order to see their relevance. If there are three nodes where one is behind another in the three, we could think that from the first node to the third one it is very important to buy the second.

The same happens with a network of friends, where users with higher interactions and products bought by them should be considered in order to provide them as a recommendation, because the influence will produce the target user to buy it sooner or later. This can be a helpful tool to find the 'influencers' of the network by (tree) level. The 'influencer' search is not that trivial, and there is a large research field in social networks for that; yet, as an approximation, for simplicity and for explainability it is better to use MST rather than less explainable computations with more complex algorithms or Deep Learning.

An item-item network can be represented in many ways; for instance, linking two items and adding the sum of total ratings where both items appear rated by the same user. But it could also add only a counter or the maximum value, or even remove nodes that do not pass a certain threshold. This will depend on the many representations that are required, which the computing resources will allow. It is important not to produce too similar representations, so as to avoid unwanted feature weighting.

There are many possible recommendations that can appear from item-item MST. An ideal example has been illustrated to better understand the insights behind MST. It is important to notice that Coca Cola is more popular than Doritos and Doritos is more popular than any jelly bean (or at least that was the idea for the illustration). Furthermore, the node of jelly bean does not contain any known brand deliberately to imagine the low relevance and high specificity of that item. Figures 3.1, 3.2, 3.3 and 3.4 show possible combinations or scenarios for a potential recommendation.

Figure 3.1, shows an item that is between two already rated items; if both items were positively rated or bought, it could be a good idea to recommend the item that is in the middle. Since this item is fully linked to the others, the user should buy it and the explanations for those recommendations are the two items and the visualization of that part of the tree.

Figure 3.2 shows an ordering where two nodes are immediately on top of an unrated node. This means that two general nodes are somehow conditioning the last node, and thus, it is recommended to buy the item if the user wants to dive more in that branch of the tree. Another example in this direction could be a collection of films that follow a progression of numbers. It is really common to

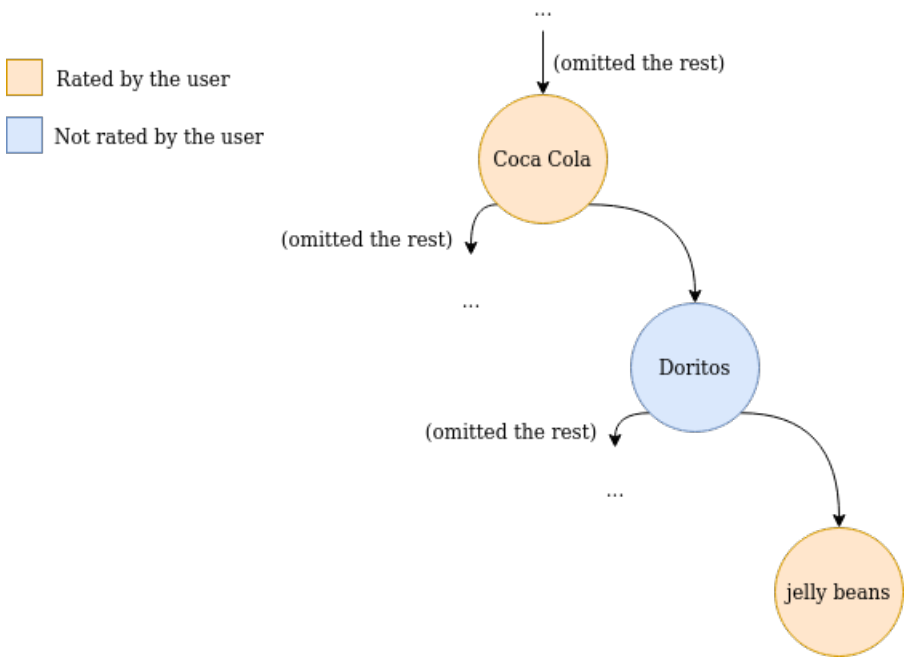

Figure 3.1: An intuitive picture of the Maximum Spanning Tree algorithm run in item-item network when the possible recommendable item is located between two rated nodes.

watch the first film and then the second, but it is odd to watch the second without having watched the first or even watching the second only.

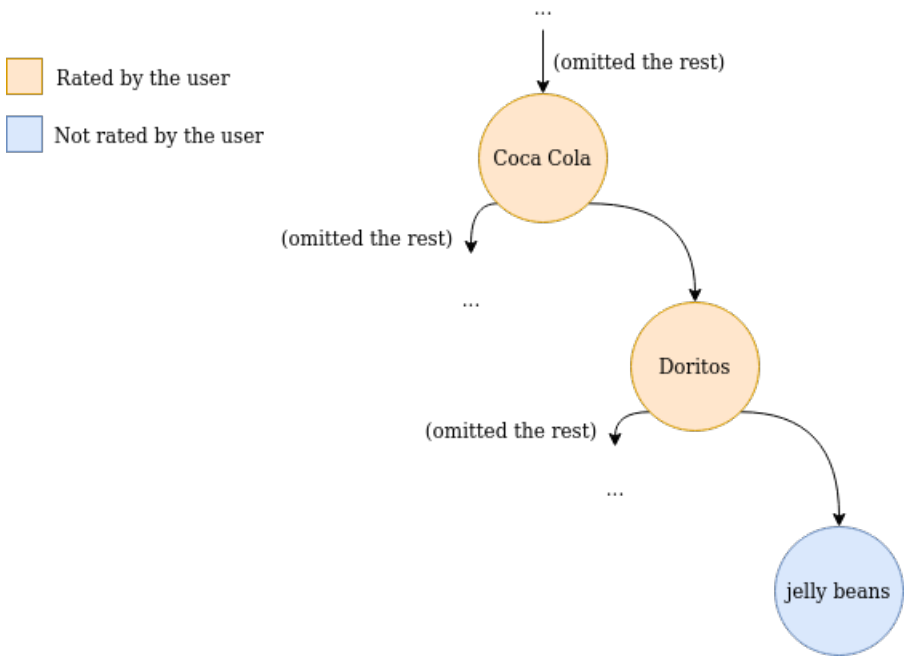

Figure 3.2: An intuitive picture of the Maximum Spanning Tree algorithm run in item-item network when the possible recommendable item is located behind two rated nodes.

Figure 3.3 shows the reverse case. It is a weird case in which the user (namely grey sheep user) has never bought Coca Cola, but since Doritos and jelly beans are bought it is very recommended to buy Coca Cola because it is a very generic recommendation that is fitting the preferences of the majority of the clients. A similar example for products sharing the same bottom-up influence characteristics could be printers and ink cartridges. While printers are bought once, ink cartridges are bought many times; if a user would have bought a printer without ink cartridges, the recommender system should strongly recommend buying ink cartridges. Therefore, even if a printer is the most important part for printing a document, it would be placed at the bottom.

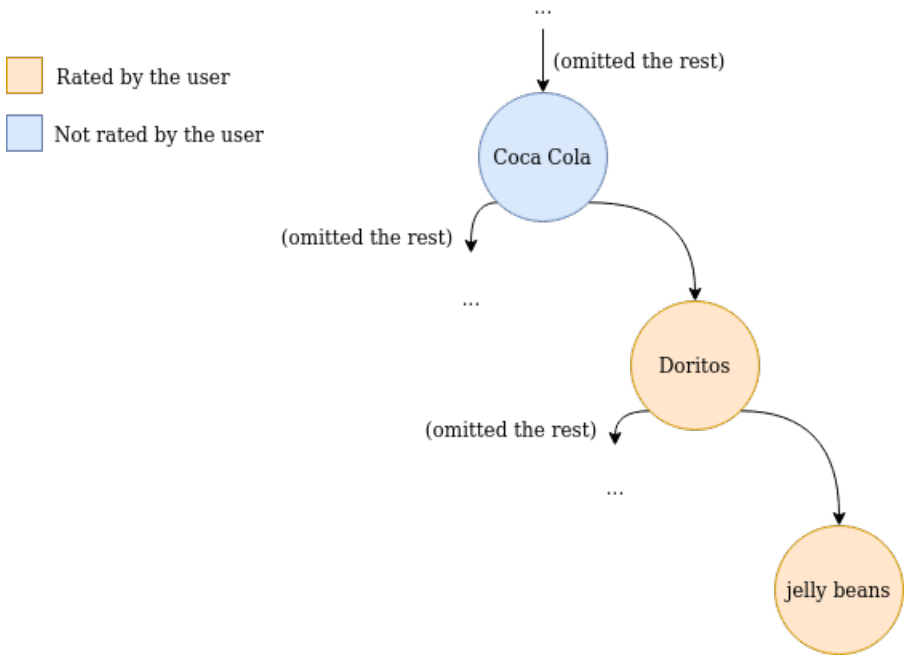

Figure 3.3: An intuitive picture of the Maximum Spanning Tree algorithm run in item-item network when the possible recommendable item is located on top of two rated nodes.

This recommender system considers recommendations in form of Figure 3.4, which is similar to Figure 3.3 without weighting the nodes that are at lower levels (jelly beans in that case). It only counts the nodes that are immediately near. It also considers Figure 3.2 and its corresponding generalization Figure 3.1, the last one counting the matches at the top and at the bottom of the target item.

Community detection group instances in bigger ones that are groups. These instances are grouped by similarity, but a similarity measure is often impossible to establish. Different Community Detection algorithms optimize certain aspects of the graph to obtain detections that are similar to a ground-truth community structure. This ground-truth could be considered subjective, because it is a way of understanding the groups of the graph.

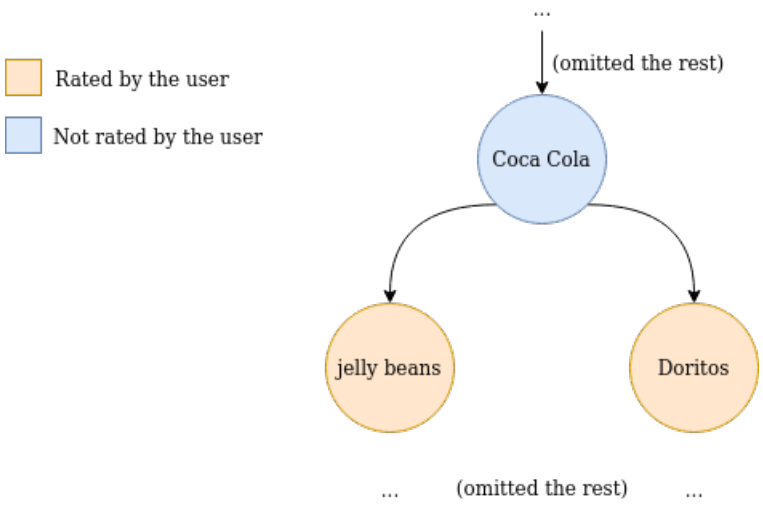

Figure 3.4: An intuitive picture of the Maximum Spanning Tree algorithm run in item-item network when the possible recommendable item is located on top of two rated nodes where both are immediate leafs of it.

In this master thesis, only the Walktrap Community Detection algorithm [31] is used, even though there are plenty of them that would let understand the groups of the network in different ways. This algorithm is based on performing random traverses across the graph from node to node. The idea behind this algorithm is that short random walks will provoke the walk not to end at a node that is far from the initial node, and thus, it is more likely to end up in the same community. After many runs, considering the probabilities, the algorithm will set the communities accordingly.

In a product-product network where products are linked and weighed, if they appear bought by the same user, strongly connected groups would appear in the same community. Then, after looking which kind of products have been bought by the user and the community of them, a recommender system could recommend in many ways; the most remarkable ones could be: recommending products from the community where the user has bought more products or recommending according to the probability assigned to the community frequency of products already bought by the user. If a user has been buying only products from one community, it should only obtain products from that community.

The intuition behind this is that users tend to perform purchases based on previous ones and that they usually buy most products of one type. For instance, a person that buys electronics also buys food and furnitures via Amazon. This algorithm would suggest electronics the most, because it is what it really likes or what is strongly linked to its job. In order to obtain recommendations of furniture compatible or linked to the one that it bought, the MST algorithm should help it with that recommendation.

In other type of networks (friend networks), detecting the community might be important in order to look up for items and check whether both users share the same passion for that community or not; in the case of sharing the same preferences, a good recommendation would be providing also a recommendation of the same community. In the opposite case, it would be interesting to provide few recommendations of that community.

Market Basket Analysis [32, 33, 34] is used by every supermarket in order to optimize the placement of products. One of the best-known techniques is the Apriori algorithm. This algorithm tries to mine rules among the list of bought items. They try to find whether one or many products imply the purchase of other products with some confidence.

With the Apriori algorithm applied to items, rules can be exploited, so that if the user has bought the product that appears on the left-hand side of the rule it will receive as recommendation the product that implies the bought product, which will be on the right-hand side of the rule. Minimum confidence barrier will determine which is the depth of the rules, whether they will be more or less generic. This can be explained as "people that buy x usually buy y".

Figure 3.5 shows an intuition for the Apriori algorithm. Although it looks very similar to Collaborative Filtering, it does not consider the relationships among the users. It only checks whether the products are bought and if one implies any other. Another example or way of representing this algorithm could be the Figure 3.6, where there is a count of rules and those that do not reach the minimum confidence are discarded. The minimum confidence is computed with the times that both items appear together divided by the times that the left-hand side of the rule appears alone.

3.1.1 Implementation

As a proof of a working system, the three algorithms have been developed in a generic form and then used in three ways not to extend the computational time. In this way, the development was faster and a big feature search and contextual explanation engineering were avoided. It is important to understand that not all datasets contain extra contextual information and, in order to test them and compare the algorithms equally, it was better to run it in a unified version of this system.

The MST [29] algorithm is computed over an item-item graph where nodes represent items and vertex between two items show that both items are bought by the same user. The weight of the vertex will add one so that they represent frequency they appear bought by all users. In other words, if two users have bought both the product A and the product B the weight of the vertex will be two. The recommendations provided by the algorithm for a user will follow the previously mentioned instructions.

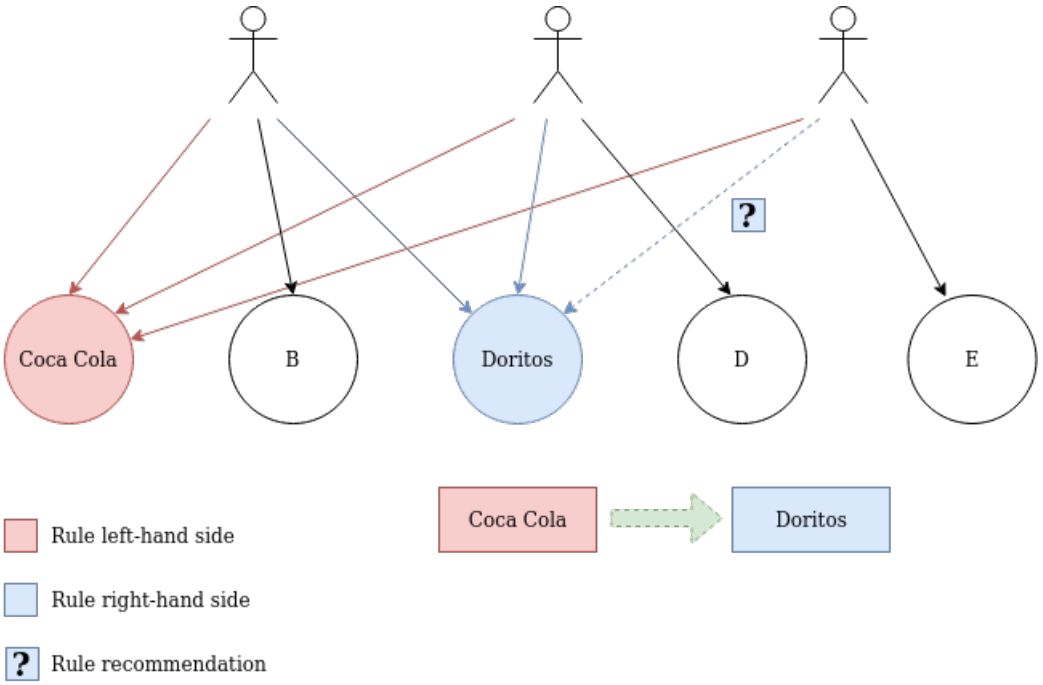

Figure 3.5: Apriori rule picture having a minimum confidence value that would activate the rule Coca Cola → Doritos.

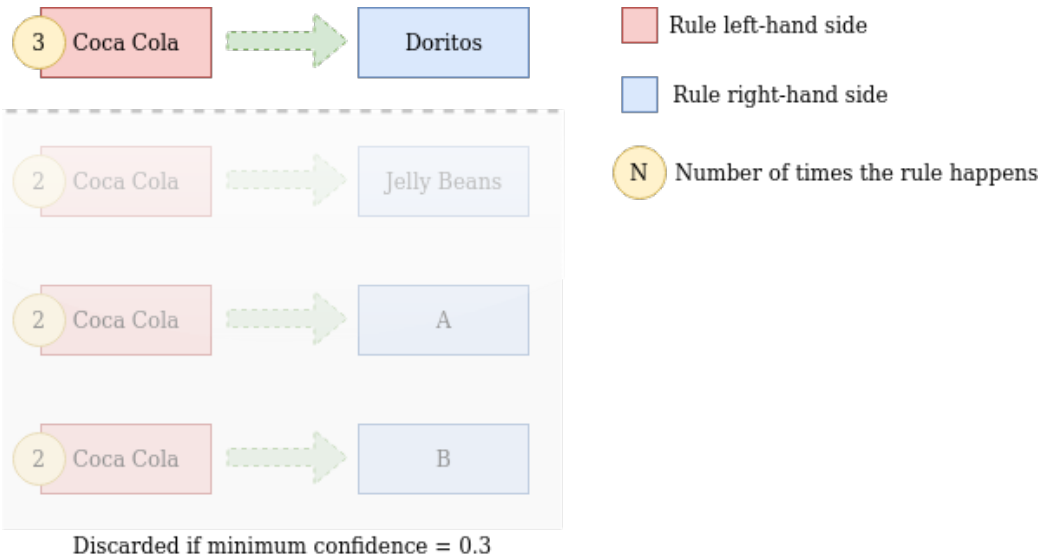

Figure 3.6: List representation of Apriori algorithm with all the examples discarded because of a minimum confidence of 0.3.

The Walktrap Community Detection algorithm [31] will work with the same graph as the MST algorithm and will produce the recommendations as mentioned above.

The Apriori [35] algorithm will take the user-item list and will produce the rules. It will subsequently generate the recommendations as mentioned above by executing them, considering what the user bought and whether there are applicable rules to those items.

3.1.2 User Parameterization

Recommendations provided by the algorithms are too many to show to the user and, in order to filter them according to the user criterion, they save some other information in each contextual recommendation database collection. MST will save the target recommendation, which are the items support that recommendation and the depth score on the tree. The Walktrap Community Detection algorithm will save the target recommendation, the community it has been assigned to and the probability they have been assigned. The Apriori algorithm will save the right- and left-hand side of the rules along with the confidence, lift and conviction.

Users should be able not only to obtain the explanation of the recommendation they are provided, but also the ability to tell the system which are their needs. For each method, there will be two parameters the user can modify, so that it can express what the user wants to receive as recommendation. The six parameters are named in a user-friendly way, with the goal of making them really usable and user-centrically designed. All the parameters have a possible range value between zero and one.

For the MST, the exploration and rating parameters will let the user balance between deep exploration of the tree leaves and exploitation of the items that are more generic. All the products have a probability of appearing as a recommendation depending on the score. If exploration parameter is high, products that are deeper will have more probability of selection. If the user sets also the rating parameter, products that have lower depth will also be more likely to appear. In this manner, the user can make an 'U'-like probability graphic if the x axis is the score, y axis is the probability and both parameters share the same value.

The recommendations based on Walktrap Community Detection let the user two parameters, randomness and feature similarity. Recommendation probabilities are maximized/minimized with the feature similarity. Recalling that probabilities are assigned according to the appearance frequency of that community in the user ratings, maximizing/minimizing the most frequent ones will be more sensible. Then, the randomization will help the user add some noise to the probabilities, so that it obtains some surprising recommendations.

Apriori recommendations let the user adapt them with popularity and correlation. Since confidence is closely related to the popularity of an item, the user can choose whether it wants more or less popular rules; correlation lets the user modify the probabilities interfering on the lift. Users will be able to find rules that are weaker or stronger and more or less popular.

3.2 Regression

The rating regression is not a relevant part of the system at all. The regression part is useful for three main tasks: showing the user the expected rating, showing

how much information do embeddings capture and, lastly, using it to sort the best recommendations and check the explainability ratio.

The rating regressor gets two inputs that are the target user's and target item's embeddings. These two inputs are enough to provide a good estimation of the rating. Figure 3.7 show how the inputs for the regressor are generated and Figure 3.8 shows the ANN architecture of it. In the first Figure, it can be seen how the sample step collects all the data from user-item candidates of train-test splits by performing a look-up on the respective embeddings and, then, joins both embeddings for the input of the ANN. The true rating is also kept for the backpropagation, for the validation and for the testing. In the second mentioned Figure, both embeddings are input, that are embeddings of six hundred features each and the intuition behind is that first with each embeddings some inference should occur, then this inference should be encoded in a smaller vector of 32 latent features (dense_2 and dense_4), then both should be concatenated, and translated to the same vector size with some computations that will allow the numbers to be merged. The dropout will regularize the network so that it does not overfit and after repeating the two last operations it reduces the size to 32 and finally to 1 that will be the output regression value.

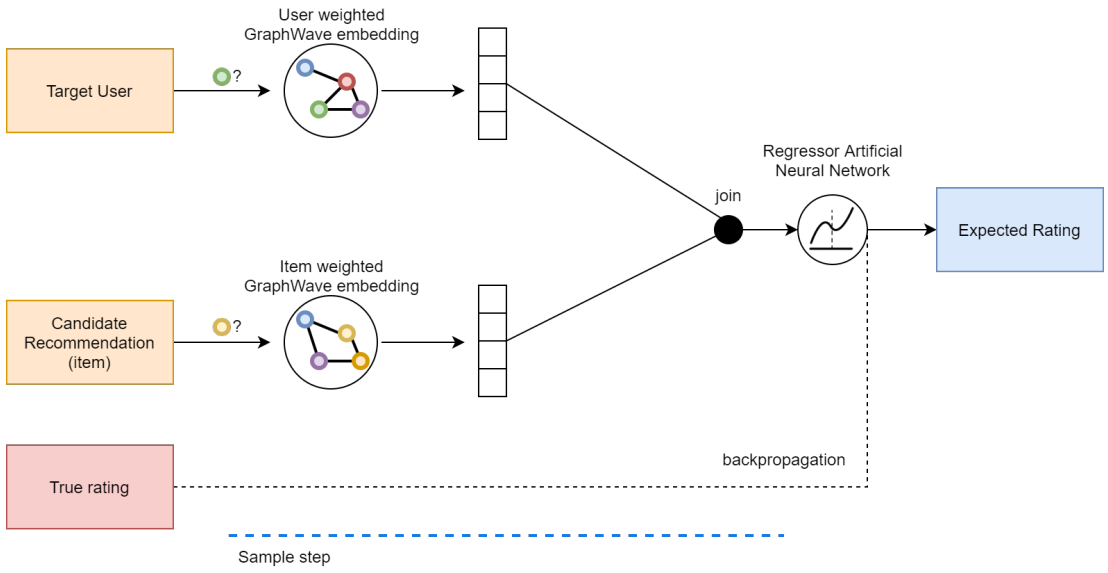

Figure 3.7: How inputs for the regressor are collected and the regressor environment overview.

In early stages of the development of this system, some contextual features that could provide more information were used. For the MST, two Boolean features appeared, asking whether the target item appeared at the MST with already rated item immediately on top or below. For the feature of the Walktrap Community Detection, a one-hot encoding of the community was included. For the Apriori algorithm, the two features selected are the number of rules where the item appeared on the left- and right-hand side. This provokes the ANN to get confused (similar to a moving target problem) owing to two main reasons. Features did not capture enough information and there there was not enough variety of samples. The other

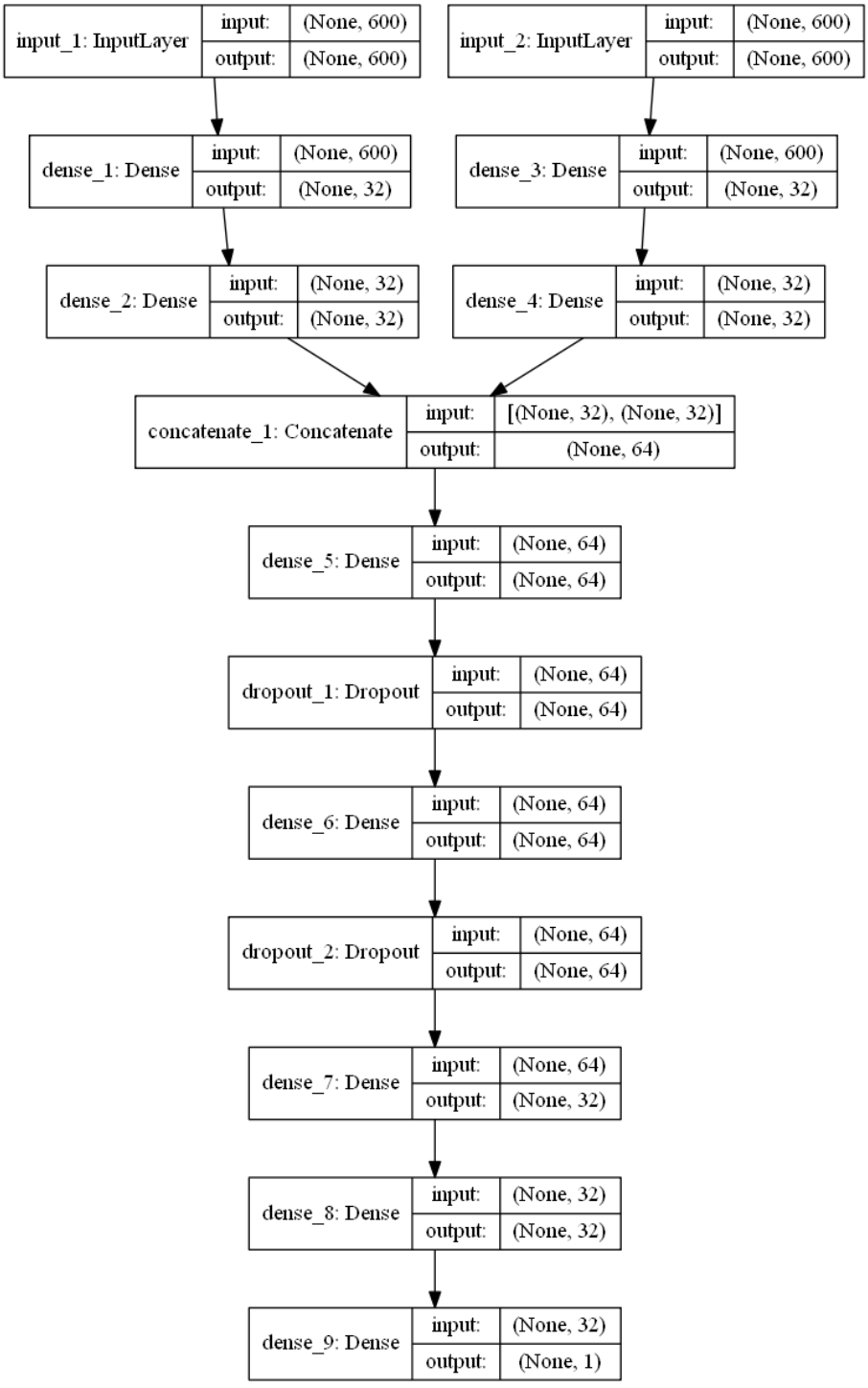

Figure 3.8: Regression Artificial Neural Network, inputs, operations and output.

reason is that, in the case of the Maximum Spanning Tree features, for the training features, the target item was compared with the already bought features without considering that item. But for the test features, all the already rated items were considered.

GraphWave is a node embedding method proposed by Donnat et al. [36]. This method tries to maximize the likelihood of nodes that share similar neighbourhood nodes, computes the graph wavelet for each node and uses that graph wavelet to compute the probability distribution of the node. The embedding is firstly the real numbers and, then, the imaginary numbers of the previous probability distribution calculus. The computational complexity is linear in relation with the number of edges. Thus, the embedding will not take too long to compute in large graphs. The graph computed for users is created by linking users with a weight of plus one when they share a product in common, whereas the graph computed for products are linked with a weight of plus one as they appear in the same purchases of users.

The network was then trained with a batch size of 128 and an Adam optimizer with a learning rate of 0.0001, first beta value of 0.9 and second beta value of 0.99, trying to learn for 70 epochs with an Early Stopping subjected to the minimum loss value in validation and a patience of 4. The validation split was 0.1, a very small value, due to the lack of samples.

3.3 Architecture

The architecture has been designed to be modular and to be suitable for any dataset, thus allowing to make high-impact modifications on demand, with no need of changes in other modules. There are new ideas for the architecture that are covered in the Future Work section.

There are three entry points or running points for this framework. The main one is related to the training of the system and testing of the regressor, the other two are related to testing the explainability and providing a recommendation for a given user.

Running the train-test is shown in Figure 3.9. The DatasetAdapter is an abstract class that contains methods that are required to implement for a new dataset. These methods are mainly deployed to load the dataset and obtain the names of the items separately. Class methods need to follow a convention for the information they return.

There is also an environment file where it is specified which configuration file to load along with the DatasetAdapter that it has to use. The configuration file specifies the information of the path, the output file names, the database and collection names, the database connection information, the minimum user ratings and the minimum product ratings (in order to filter them). The environment file also allows to modify the train-test ratio and establish whether it has to recompute everything or not. There are also parameters that bear relation to the results size the users are provided.

The build step filters the users and products that do not meet the requirements specified in the configuration. Ratings are scaled between 0 and 1 and a train and

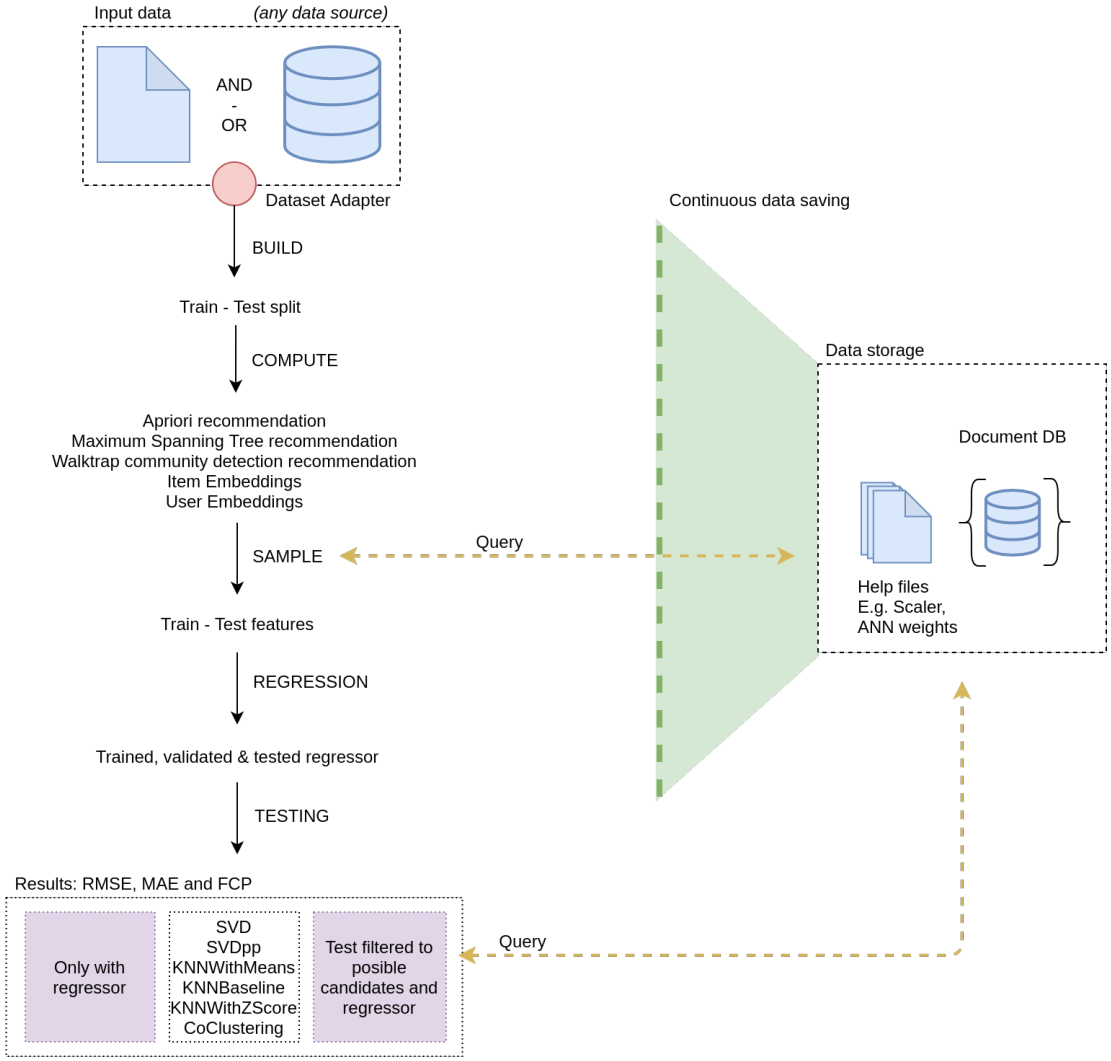

Figure 3.9: Current Architecture.

a test split are conducted where there is a minimum size of same product and same user at train and test. For sanity check, the system will check whether at train and test there are the same users and products; in other words, whether at least the test split is a subset of both user and products of train split.

As an extra feature, a bar chart of products and users and respective ratings size are computed and output to the path specified in the configuration file.

The compute step computes all the required information in the following order. First of all, the Apriori rules and the recommendations for all users. Then, the product weighed graph that is necessary for the next computations, the MST algorithm and the Walktrap Community Detection and their respective recommendations; the item embeddings that are immediately after computed also need that graph. Finally, the user graph is computed, and the user embeddings as well. In order not to overload the Random Access Memory, the graphs that are generated are removed from memory once all the computations that need the graphs are terminated.

The sample step, as mentioned above, will look up for each group of user-item rating the corresponding embeddings. User-item embeddings will be piled in a matrix and the rating in a vertical vector.

Then the ANN will be trained with the parameters mentioned (user embeddings and item embeddings as input and the true value of the regression for the backpropagation) and the results will be stored in a matrix.

In the testing phase, the regressor will be compared with other methods and the results will be shown in the console.

Continuous saving will store at the database the train split, the algorithms and their recommendations. It will also save in files the sampler (with the scaler and other information), the train and test data, the MST and the regressor with its structure and weights.

3.4 Explaining Recommendations

The process for explaining recommendations starts when user inputs parameters and demands recommendation. The recommender system will then sort the recommendations depending on the user input, maximizing the expected rating or maximizing the explainability. It is worth recalling that, even if the user wants to maximize the expected rating, at least one explanation is required for any item to appear as they are sampled from the database and all the recommendations are targeted with user ID and product ID.

The simplicity of the recommendation is what lets any kind of user understand the recommendation it is provided. The natural language used to express the recommendation is a simple template, but still valid, and if the user selects to optimize the explainability, it will get as many explanations as matches are in the sampled subset. It is important to understand that because of the user parameterization, even if there are N matches on the database search, the user will get from 1 to N recommendations, as they can appear in the sampled subset or not. This strategy makes sense because the user inputs should be understood by the user as valid and useful inputs. And if the sampler that is modulated with that parameters would make the trick to look up for all the matching explanations, the user would lose a part of the importance that it is giving to the parameters. It is focused on a kind of hidden feedback, not to output always all the recommendations for the item.

As an example to what is mentioned above, if a user selects some parameters, and the results size per contextual recommendation is 50 and the final recommendation size is 7, the system will try to match those 150 recommendations (considering that there are 3 contextual recommenders and a limit of 50 per recommender; 50x3), will then sort according to what the user specified (expected rating or explainability) and the explanations will be generated.

Some recommendation examples are provided below in order to check how recommendations are explained with both text and images. Three examples where recommendations were separately provided by algorithms for different datasets are analyzed; in one of them two different explanations from algorithms are given.

The first explanation is provided for a user in the LastFM dataset with the following text: *'MST items have been found for this recommendation. The Beatles was suggested by a product that is Radiohead the depth in the tree is 49. The Beatles was suggested by a product that is Simon & Garfunkel the depth in the tree is 154. The Beatles was suggested by a product that is Kings of Convenience the depth in the tree is 171.'*; Figure 3.10 shows the tree explaining the same in a more visual way. As can be observed, Radiohead is in other cluster of artists, although it seems to be derived from The Beatles because of the listening influence.

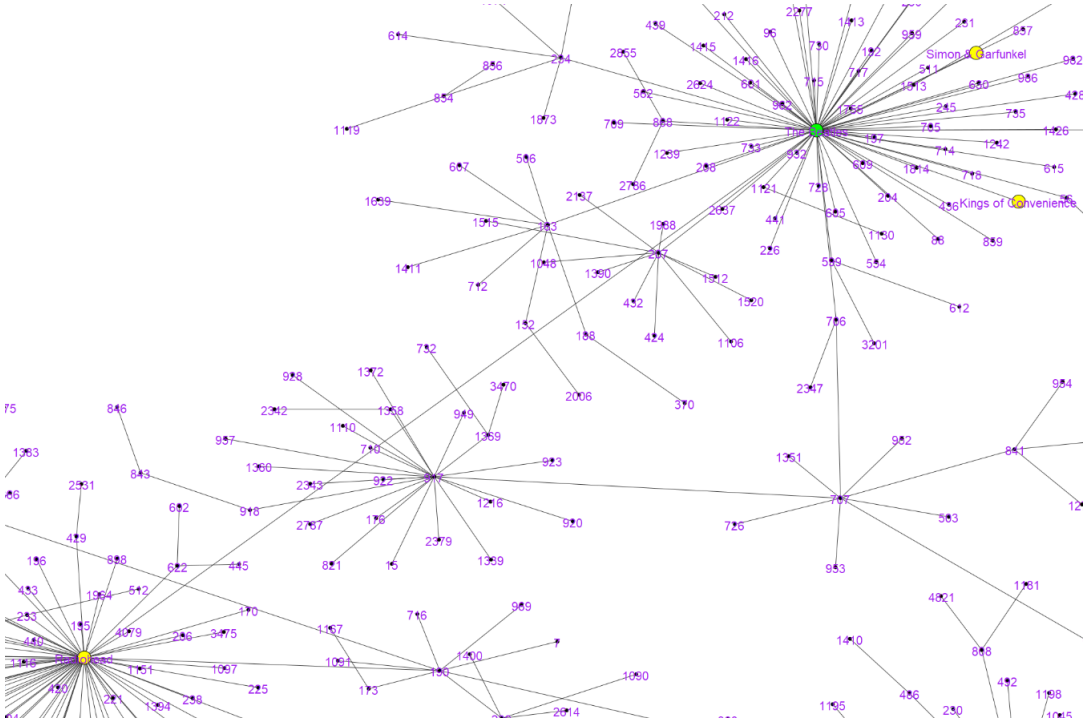

Figure 3.10: Maximum Spanning Tree of LastFM showing Radiohead on bottom left in yellow color forming the core of a tree, connected to The Beatles (the provided recommendation), on top right, in green color; and Simon & Garfunkel and Kings of Convenience are in yellow leaves of the core tree that are The Beatles. Three yellow nodes support directly the recommendation of The Beatles for the user, as there were artists listened by the user.

The next recommendation is given for a user in the MovieLens 20M dataset. The recommendation is activated by the Walktrap Community Detection and the MST algorithm are the resulting ones: *'Walktrap items have been found for this recommendation. Lake Placid (1999) was suggested because it belongs to the community nº 0 this product has a probability of 0.02127'*, *'MST items have been found for this recommendation. Lake Placid (1999) was suggested by a product that is Braveheart*

(1995) the depth in the tree is 79. and Figures 3.11 and 3.12 are provided. In both films Brendan Glenson performed as an actor, and this can be one of the reasons why people watch them together and appear in the MST. Because of the relations of the mentioned films with other films, both share the same community ID.

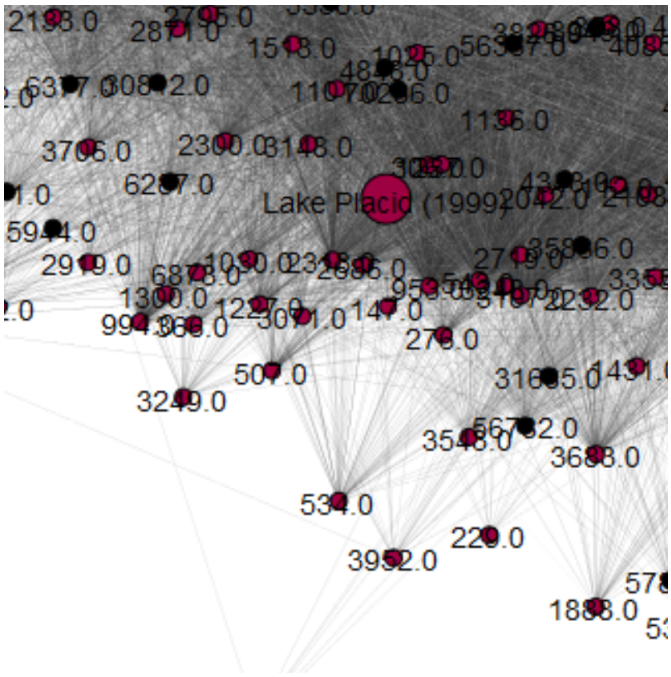

Figure 3.11: Walktrap recommendation in MovieLens 20M of LakePlacid. Two communities can be differentiated with colors in this picture. The target community appears in maroon color.

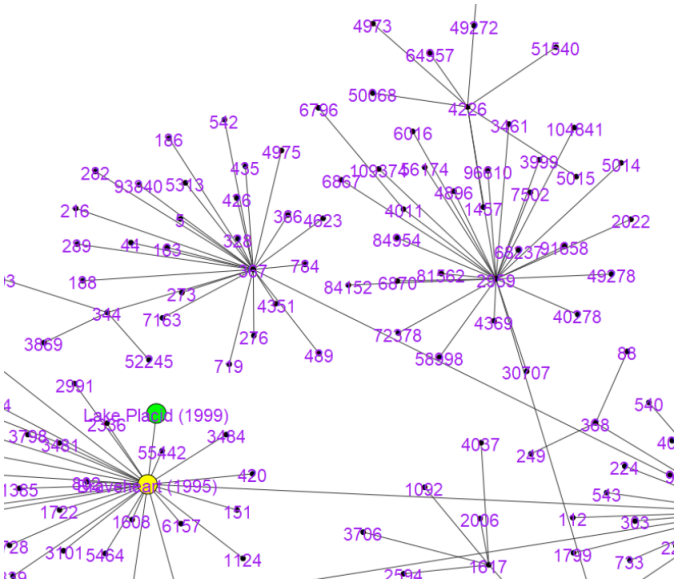

Figure 3.12: MST recommendation in Movielens 20M of LakePlacid in green color influenced by the already watched film named Braveheart in yellow color.

This recommendation is provided for a user of the MovieLens 20K dataset. Many Apriori rules with high confidence are fired and the explanation is the following: *'Association rules have been fired for this recommendation. With a confidence of 0.5154 The Mask, Indiana Jones and the Temple of Doom imply Indiana Jones and the Last Crusade. With a confidence of 0.5420 The Silence of the Lambs, Indiana Jones and the Temple of Doom imply Indiana Jones and the Last Crusade. With a confidence of 0.5514 Independence Day, Indiana Jones and the Temple of Doom imply Indiana Jones and the Last Crusade. With a confidence of 0.5164 Gattaca, Pirates of the Caribbean: The Curse of the Black Pearl imply Indiana Jones and the Last Crusade.'*. There is no picture accompanying this explanation because rules do not have a way of real visualization. It has to be remarked that one film of Indiana Jones implies another film of the same collection of films as expected; this happens many times in combined rules with the mentioned film. Pirates of Caribbean is also a well-known contemporary film saga sharing the same popularity.

Recommendations provided can be used for platform owners or even for users. Platform owners can understand how the whole recommendation space is formed, how products are linked and how recommendations are provided for users. In case they adapt slightly the explanations shown above, users could get understandable recommendations. Users should not be aware of the algorithms used to provide recommendations but at least get some insights about them. As it is mentioned above, rules could be explained with the influence, for instance, *'People that watched The Mask and Indiana Jones and the Temple of Doom also watched Indiana Jones and the Last Crusade'*; for the MST recommendations the explanation should be *'The Beatles was suggested by an artist that is Radiohead which is very generic, by an artist that is Simon & Garfunkel which is less generic and the artist Kings of Convenience that is more specific' (attaching the picture).* Notice that some words have been adapted to the domain, for instance the last example changed from 'product' to 'artist'. This is owing to the fact that the explanations were templates trying to show as maximum information as possible and the user adaptation was obviated. For the MST or for the association rules, the respective depth (specificity) and confidence could also be displayed with gradient colors in a UX-friendly way.

3.5 Customization

This system aims for complete customization, since it can be adapted to any dataset and deployment scenario. At the current development point, in order to be an online recommender system, it is required to run the system at least once every two days to generate all the recommendations. Services should point to the old database and, once the new one is generated, the services should point to that one. It should be noted that the proposal of this thesis is not providing any solution to the Cold Start and, thus, new users and products will be skipped unless they pass the minimum ratings specified in the configuration file. In the next subsections, it is explained how to handle with the Cold Start problem in other more interesting ways.

3.5.1 Regression

The regression part can also be modified to include more information for the regressor to better approximate or predict the expected rating of an item. Taking embeddings from other context graphs can yield postive results. For instance, computing friend user-user network embeddings could enhance the results of the regressor.

Probably, including similar graphs but built in different ways could also improve the results if the regularization is tuned, so that it prevents overfitting.

As explained above, adding hand-crafted features does not seem to be providing positive results at all; yet, probably, including user data (genre, age, country) and including information of items (release date, category, price) could improve those results. It is important to notice that adding more and more features will not improve the results, unless there is enough data to cover all the space. In other words, model complexity will increase when adding more and more features and, if there is not enough data to show the impact that all features have on the final regression, it can yield worse results than only using embeddings.

3.5.2 Other Datasets

In order to add other datasets, the first step is to make a subclass of the DatasetAdapter and then specify how the whole dataset should be loaded. Then, the configuration file should be changed along with the environment file, so that the desired requirements are met. The DatasetAdapter expects some columns to be presented with a certain name, hence column renamings are necessary.

In case there is other information that should be loaded (friends, product specifications or user data), it might be a good idea to add the methods at the DatasetAdapter subclass. Then, if there are new specifications of how the system should make the train test split or how the items should be discarded, it is a great idea to modify the build step.

Compute step has two embedding algorithms implemented that are node2vec [37] and GraphWave [36]. Embedding algorithm can be switched along with how the graphs are generated; if other contextual information needs to be computed or an extra embeddings are wanted for the regressor, they can be included and modified there.

If new information is wanted to be added into the regressor, both the sampling part and the regressor should be changed. The sampling will have to look up with the DatasetAdapter or any utility function to get that extra information.

Adding more contextual method recommendations should be done integrally, since the testing step will check for all the recommendations in order to compare the error with other recommender algorithms. Also for the user recommendation, the

file should be modified to include the recommendation of that context. Contextual algorithms have the same methods implemented. It is important to emulate the behavior with new contexts because it is a way of checking they are implemented correctly.

3.5.3 Cold Start

In production Recommender Systems, when a new item is added to the catalogue or a new user is registered, there is an issue on how to recommend the newly added product and how to recommend products to a new user about which we do not know anything. This cold start problem is not tackled in this master thesis, but two compatible and interesting options are provided in order to make this system production ready.

For a new user in the system, it is strongly recommended to provide very well-known items to the user as recommendations, instead of running the system with the user, which would result in empty recommendations. A convenient option could be using the same method as Netflix: providing at the registration process of the user a basic form where the user can select which films are to its taste in order to run the recommender system. For that selection, it could be interesting to add items from different levels of the Maximum Spanning Tree algorithm or from different communities of the Walktrap Community Detection algorithm. This would let the system initialize a given user in many different levels and not only with popular products. When the user selects the items, those selections could be stored in the database with a rating value that could be the mean of the range of the possible values that the system allows. That information should be stored as a flag that would mean that it is not a real rating. A trigger on the database would be useful to check whether the user has surpassed the minimum real ratings in order to remove the ones that were saved during the initialization selection in registration.

New products are more difficult to integrate in the system, but there are many ideas that could help solving that problem. The first idea would be adding always one newly added product to recommendations provided to the user with the explanation that the product is new. The second idea is that, since adding new products is always or very often done by humans, it could be a great plan to let the user select among randomly chosen products to which product is more similar. As a side effect, it could be helpful to build an user-annotated similarity graph, as it could lead to new research opportunities in node similarity fields learning the distribution of humans that will agree or disagree, which a task that is very known in Natural Language Processing [38]. With a certain probability, when recommending products that are already rated, looking up to that similarity graph and replacing them could be a great idea to start including them in the recommender system. The last way to include a new product in the recommender system is far from recommender systems' scope of research; it is rather a User Experience (UX) and some business intelligence that is to promote new products on the main page or in separated a section, like Netflix does.

Chapter 4

Evaluation

4.1 Methodology

Knowing the performance of the proposed recommendation system works is crucial to understand how it will behave in a real-world scenario. It is important to highlight that the model is not trying to fit the user's likes, but rather consider the user's likes and ethically[1] [39] let the user interact with the system. In this case, it is very difficult to detect whether the system behaves well or not. System performance has not been measured in the time dimension because the system is big and there is some sub-optimal code. Moreover, this system is thought to be deployed and thus all the information is stored and queried, which adds more latency to all the operations.

There are two blocks of hypothesis to validate the recommender system is behaving well. The first one is mostly related to the regression, but also to how the recommendations are generated and how valid the recommendations are. The second block is fully focused on validating the explainability of the model.

It is important to consider that the proposed method does not perform recommendations with a very complex system and thus the recommender systems with which it has to be compared should have similar simplicity. To test the regressor, features (user-item embeddings) from test rows are obtained and the regressor is performed. Then the results are compared with other methods.

The proposed method is compared with: the Single Value Decomposition (SVD) method, the improvement of the method SVD (namely, SVDpp), K-Nearest Neighbours (KNN), KMeans, an efficient computation of the KNN with Z score and, lastly, Co-Clustering based on Collaborative Filtering algorithm [40] (all these using Surprise library[41]). These methods are less complex than the one proposed, the recommendations/regressions obtained from these items are mathematically derived and thus they will have some accuracy, but the method proposed regresses with an ANN and the recommendations are obtained heuristically. Hence, they are thought to be equivalent, if not superior, for testing with them.

[1] Addictiveness and extreme content (https://towardsdatascience.com/the-ethical-and-privacy-issues-of-recommendation-engines-on-media-platforms-9bea7bcb0abc).

The system is compared with two very known rating-quality metrics and another one that is not so known, respectively: Rooted Mean Squared Error (RMSE), Mean Absolute Error (MAE), and the last one, Fraction of Concordant Pairs (FCP). FCP was popularized by Koren et al. [6]; as stated in Koren et al., FCP measures the proportion of well ranked items, a generalization of Area Under the Curve (AUC) metric.

H1. The first simplest hypothesis is to check whether the regressions are positive with the small information provided (only simple embeddings). It is not expected for this system to overcome the other methods but at least to get close in the results.

H2. The second hypothesis is more complicated and has never been seen in any paper in Recommender Systems. If the proposed method would not provide some recommendations that appear in the test data, the system should not behave well with those ratings. Thus, it is interesting to filter those recommendations to compare again and observe which has been the change. The results could be interpreted in two ways: if they improve, the system can predict better with candidates derived from contextual information; if they keep the same, the recommendations proposed are not bad at all and they keep in the same line.

The third and fourth hypotheses are similar in the computation of results, as the two previous hypotheses were. Both hypotheses will use an offline explainability Model Fidelity ranking-quality ratio metric proposed by Peake et al. [42]. This Model Fidelity is computed as the division between the recommended items that are explainable and all the recommended items.

H3. The third hypothesis continues using the test data and it is thought that, if user selected items that appear in the test data split can be mostly explained by the current recommender system, the system is performing well not only on the recommendations, because it could provide them, but also on the explainability. Then, the Model Fidelity is computed with the items that appear in test data that are explainable divided by the items that appear in the test data.

H4. The fourth and last hypothesis is that, even though the recommender system works with the user by exploring the possible recommendations generated by the system, the system could also be used for any kind of recommender system, maintaining a great part of the explainability, as well as taking advantage of the capabilities of other recommender systems. Yet, although it can be considered a good idea, it is actually losing the ability that the user has in order to get the recommendation via parameterization and thus it can be less ethical, unless the other recommender engine lets the user interact with it in an open way. For that purpose, the trained regressor is used to get all user-product expected ratings and provide a Top-N recommendation with N equals to 2, 4, 6, 8, 14 and 20 and then use the Model Fidelity formula to check whether there is an explanation for those recommendations or not.

With this hypotheses and both ranking- and rating-quality metrics proposed a depth analysis is assured and thus a complete evaluation of the proposed solution is provided.

4.2 Datasets

The system proposed in this master thesis is tested with many datasets because each dataset has different characteristics and follows different distributions. Furthermore, it is relevant to do so in order to check if the system works as a generic system and not only for a single recommendation scenario.

Because of computational resources, some datasets have been reduced through random sampling. This could raise a question on why to use those datasets. As explained above, datasets are of different types. There are datasets of movies, a dataset of songs, a dataset of books, a dataset of foods and a dataset of jokes. It can be thought that each dataset has its own characteristics and the users do not interact in the same way. This data reduction can be useful to see how the recommender system behaves with missing data/ratings. For training and testing, after all the data reduction or sampling, all datasets have been split 80 percent for training (10 percent of it for regression validation) and 20 percent for testing.

The differences among datasets are large not only owing to their size, but also to the fact that the distributions of ratings and interactions are different because there are many factors of influence (for instance, a platform, the year of collection or the collection mechanism). And inside those factors there are many (underlying) sub-factors; for example, the platform's UX can make some users interact with the platform or not, and the same UX can influence what the user provides as rating, even the time in which the platform is used (sports time, relax time, lunch/dinner time), the main group of users that are involved (readers, cinephiles, etc.), and so on.

4.2.1 LastFM

LastFM[2] is a platform for listening to online music, like Spotify[3] and SoundCloud[4]. A big dataset to research in recommender systems adapted to music recommendations was provided, together with much contextual information (combination of user, artist and number of plays; user friend relationships and artist tags). Yet, for simplicity, only triplets are used to generate different contexts or perspectives of the same data and provide recommendations.

There is a wide variety of subsets of this dataset. The one selected is from HetRec2011, which was the International Workshop on Information Heterogeneity and Fusion in Recommender Systems held in 2011. This dataset in particular has 92.800

[2]https://www.last.fm/
[3]https://www.spotify.com/
[4]https://soundcloud.com/

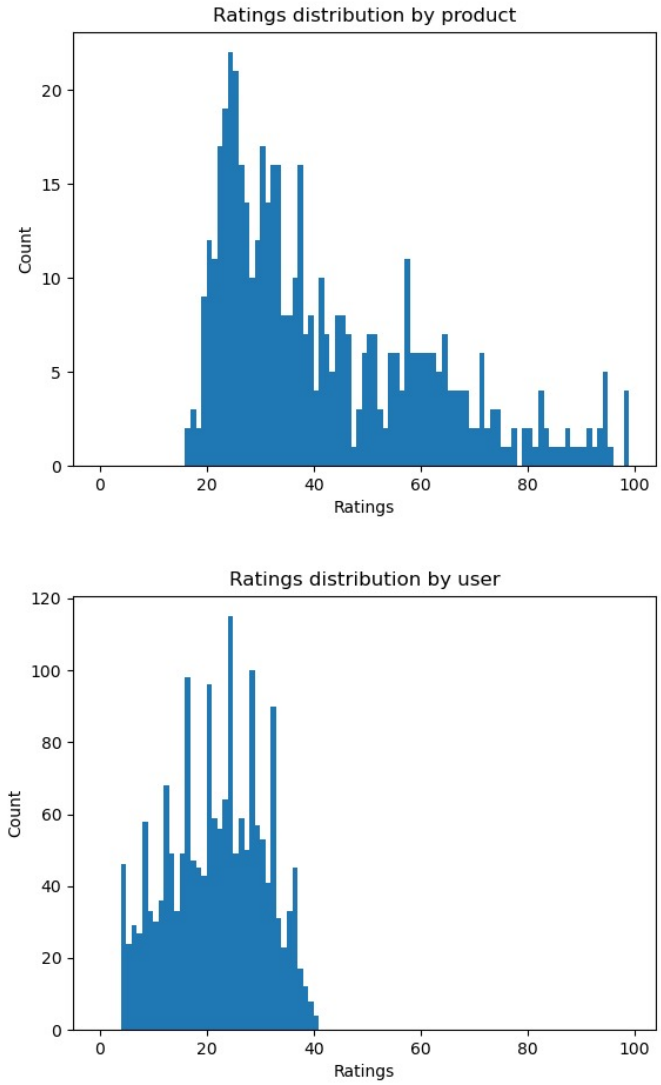

Figure 4.1: LastFM dataset distribution of train data.

artist listening records from 1892 users. No explicit reduction was performed, the minimum required product ratings were 25, the minimum user ratings were 20 and the minimum confidence set for Apriori rules was 0.1. Resulting distributions are displayed in Figure 4.1, clearly showing a mean in the twenties for user rating distribution and in the forties for product rating distribution.

4.2.2 Sushi

Introduced by Kamishima et al. [43] in a work about an efficient clustering technique, this dataset contains user surveys of sushi preferences. There is much contextual information available about both users (age, gender, time to fill questionnaire, Japanese region, etcetera) and items (style, major and minor groups, oiliness, prize).

There are many versions available that are subsets with combinations of users and items; the one selected is the biggest one, containing 5.000 users and 99 items

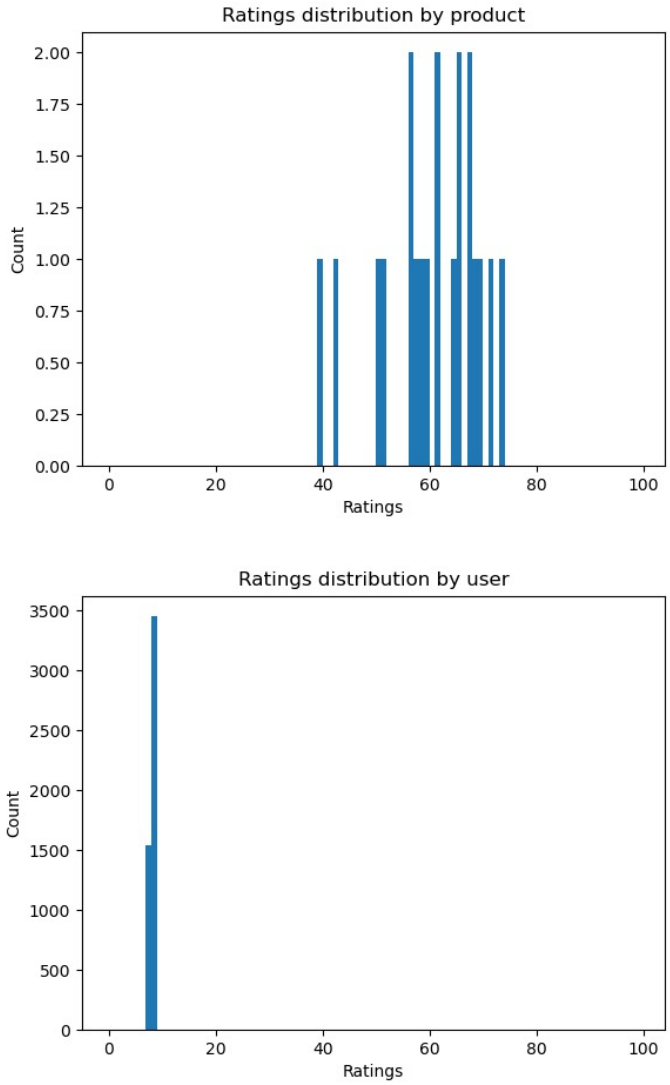

Figure 4.2: Sushi dataset distribution of train data.

in total. No reduction was performed, and the minimum confidence set for Apriori rules was 0.0005; this is because it was very difficult to mine rules inside a limited set of products inside the survey. Fortunately, the users could express that they had never tried the item by replying "-1". Then the context rules in this case were showing which sushi foods where more popular, implicitly considering the Japanese regions. Resulting distributions are displayed in Figure 4.2 showing a very noisy chart, because this dataset is obtained from a survey. User ratings can look weird because this is the distribution of the train split and recall in which users could reply "-1" to an untasted product.

4.2.3 Movie Tweetings

Movie Tweetings, as its name indicates, is a dataset extracted from film-related tweets of the social network Twitter. It is a dataset created by Dooms et al. [44] that aimed to explore the extraction of ratings from the social network mentioned.

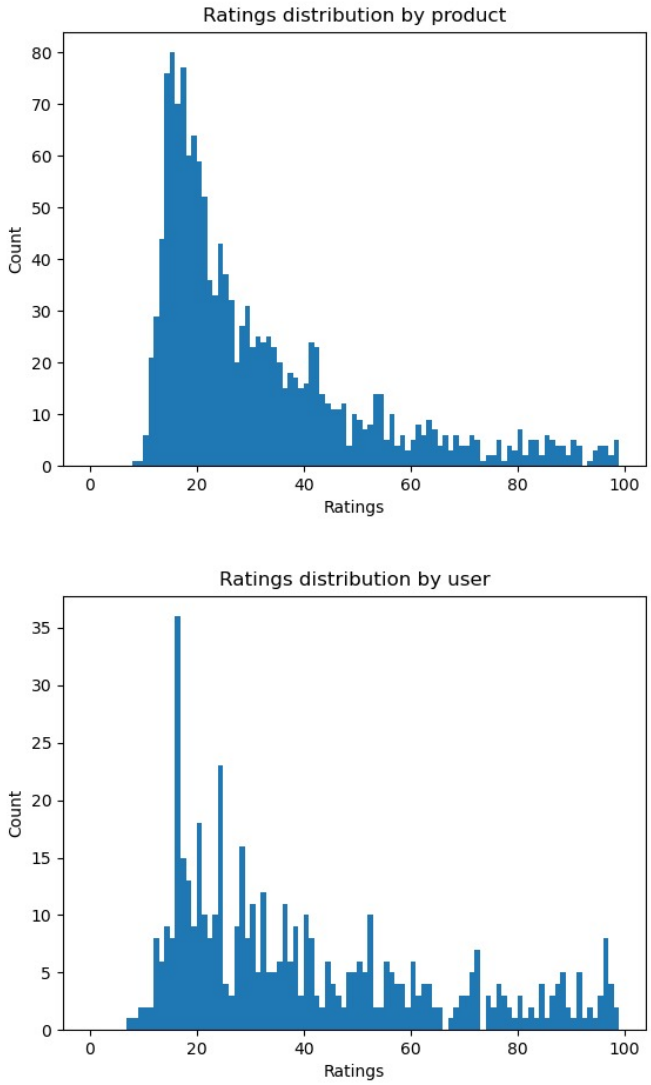

Figure 4.3: Movie Tweetings dataset distribution of train data.

Authors of the dataset utilize a script that is running all the time capturing those tweets and including them into the dataset. Thus, the dataset is a constantly growing dataset, whose statistics are persistently on the increase as well. This could also be interesting and useful for scientists who want to work on an online recommender system that it is growing every day but who do not work in a company that has this data. All the extracted tweets are structured, since they are automatically published when users rate a film at IMDb [5] (Internet Movie Database, a website). This will occur if they have their Twitter account linked and they allow automatic tweets.

The dataset contains 100.004 ratings and it is a subset of the current dataset; it is filtered to 15 minimum product ratings and 12 minimum user ratings without explicit previous data reduction. The minimum confidence for the Apriori rules was 0.08. Resulting distributions are displayed in Figure 4.3 showing two aligned bar charts

[5]https://www.imdb.com/

in both user and product rating distributions, even though the user distribution is, to a limited extent, irregular.

4.2.4 MovieLens

The MovieLens dataset is developed by researchers at the University of Minnesota. It is similar to the Movie Tweetings dataset in the sense that it is a continuously growing dataset. The reason for this is that there is an online-hosted recommender system called MovieLens where users can rate movies and obtain recommendations accordingly.

Two subsets are used in this master thesis. The first one contains over 855 thousand ratings and the second one, 20 million ratings. The first dataset is obtained from the HetRec2011 conference and the second one, from the MovieLens web page. The first dataset is reduced to the 60 percent and then filtered to set as required minimum product ratings of 20 and minimum user ratings of 25; the minimum confidence set was 0.02. The second dataset is reduced to the 20 percent, records of minimum product ratings, to 50 and minimum user ratings, to 20; the minimum confidence for Apriori was 0.002. Resulting distributions are shown in Figures 4.4 and 4.5 from the MovieLens20K and MovieLens20M, respectively; the first distributions show a step hill on the product rating chart and the distribution of user appears to be very noisy, whereas in the second distributions the product slope is mountain-like, in the contrary part of the distribution of users, which a is steep hill complementing "the other part".

4.2.5 Books Crossing

The dataset crafted by Ziegler et al. [45] for a recommender system approaching topic diversification contains ratings of books by users, concretely, 278.858 users, 271.379 books and a total of 1.149.780 ratings.

The dataset is not sampled, minimum product ratings are 15, minimum user ratings are 12 and the minimum confidence for Apriori rules is 0.05. Resulting distributions are displayed in Figure 4.6 showing two almost aligned bar charts in both user and product rating distributions, although the user distribution slightly irregular and goes from high to low while the products distribution is a steep hill, more steep from down to high and then it softens from high to low.

4.2.6 Jester

Goldberg et al. [46] introduce this method in their work about Collaborative Filtering. The dataset deals with ratings of jokes that appeared on a Jester website. There are three current versions of this dataset, each of the versions has been collected in different date intervals: the first (1) dataset was collected between April 1999 and May 2003; the second (3) dataset, between November 2006 and March 2015; and the third (4) one, between April 2015 and November 2019. There are more versions of the dataset, but the authors decided to merge the two versions of

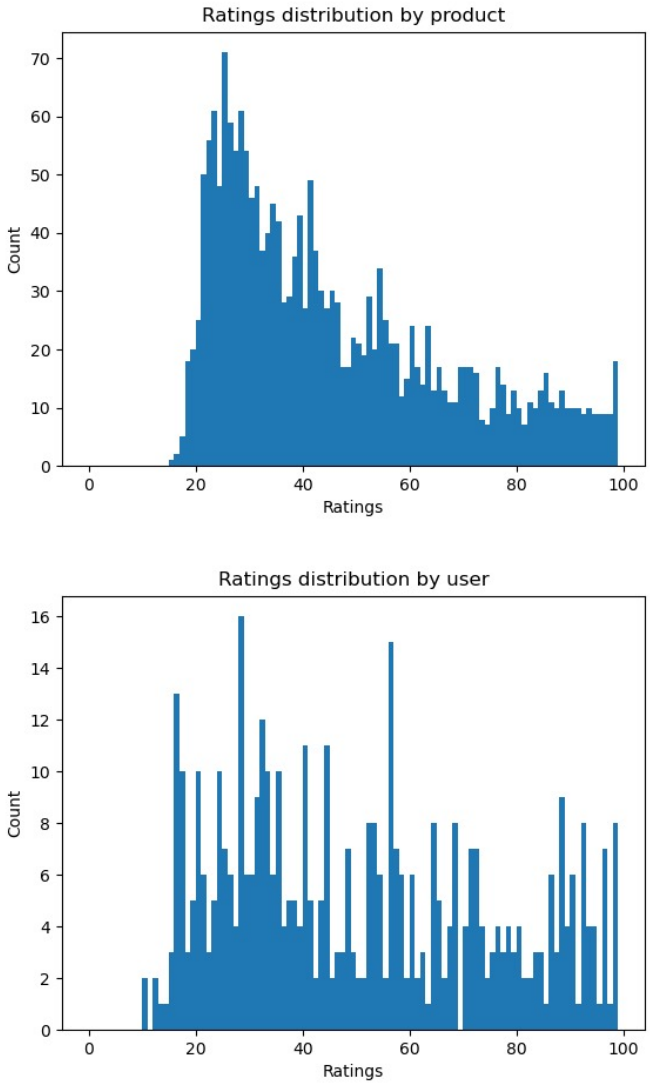

Figure 4.4: MovieLens 20K dataset distribution of train data.

the second into the third and created a fourth dataset with new data. This dataset is also interesting because it evolves over time, like previous ones.

This dataset is complicated for the proposal, since the connectivity among items and users is large, and the weights are large as well. The part of the system that experiences some difficulties reflected in the computing time is the embedding part. The rest of the components and steps work adequately because the dataset is not exceedingly large at all.

The first dataset is split into three subsets; these three subsets have been used in order to check how different Jester versions of the same dates work with different information. The first split contains 24.983 users who have rated 36 jokes or more out of 101 jokes, the second split contains 23.500 users who have rated 36 jokes or more out of 101 jokes and the third split contains 24.938 users who have rated between 15 and 36 jokes out of 101 jokes. All the datasets are sampled to the 60 percent because

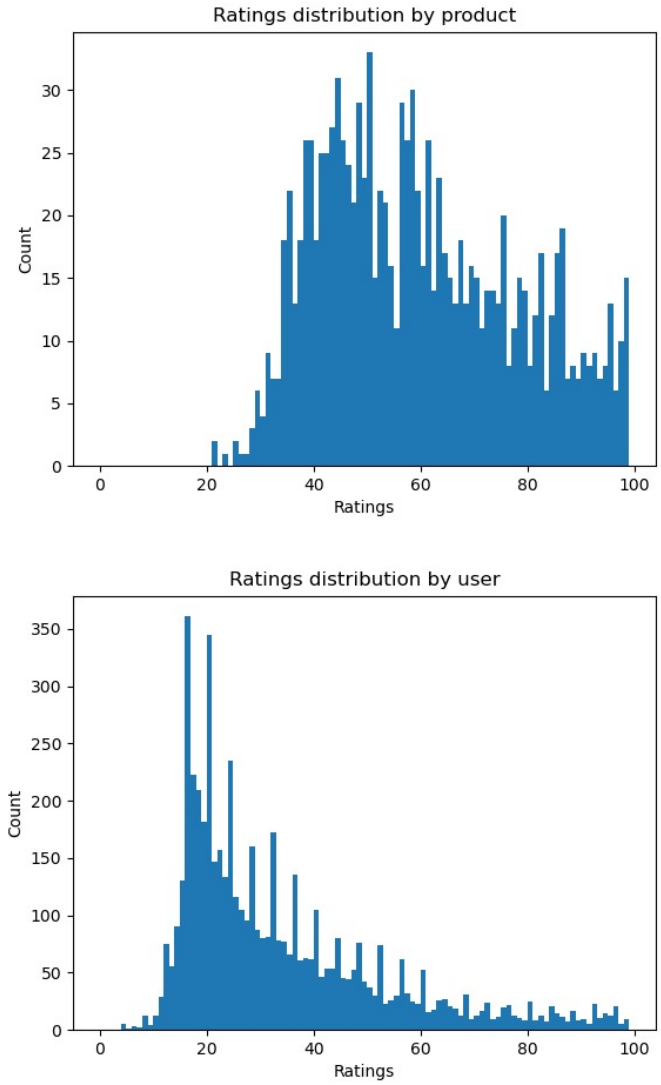

Figure 4.5: MovieLens 20M dataset distribution of train data.

they are large. The minimum product rating is 45 and the minimum user rating is 30, with a minimum confidence for Apriori Rule Mining of 0.1 for Jester 1 and 2, while in Jester 3 the values are 20, 7 and 0.02, respectively. Figures 4.7, 4.8 and 4.9 show the distribution of products and users; Figures from Jester 1 and Jester 2 are similar, with two maximum points in users' distribution, and Jester 3 has also a similar shape, more compressed, though. As regards products distribution, Jester 1 and Jester 2 have the same unique bar at 150, whereas in Jester 3 there are a few more bars distributed near in the center of the chart.

4.3 Hardware

For this work, a desktop computer has been used, the main characteristics of the device being: 64 gigabytes of Random Access Memory, an Nvidia RTX2080Ti graphics card with 11 gigabytes of video memory GDDR5, an AMD Ryzen 9 3900X

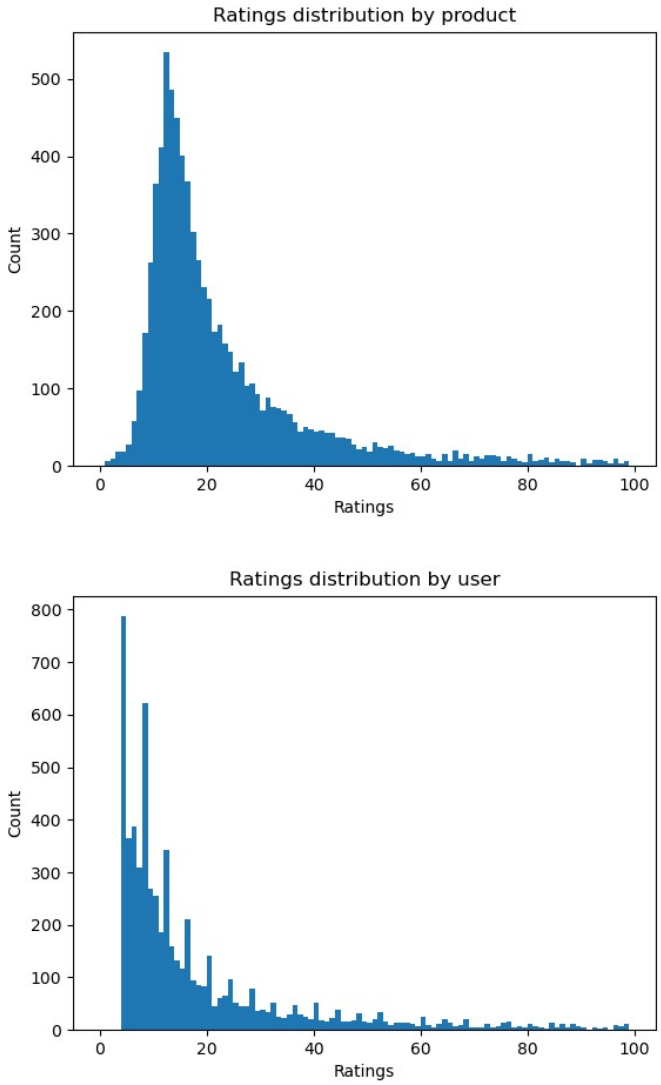

Figure 4.6: Books Crossing dataset distribution of train data.

12-core processor with 3.8GHz base clock speed and a Toshiba-RC500 NVMe M.2 SSD with 1700 megabytes per second of read speed and 1600 megabytes per second of write speed. All the final tests have been run with this computer.

Older desktop computers can also work smoothly with this recommender system. In fact, it has also been tested with a computer with 32 gigabytes of Random Access Memory, a Nvidia GTX680 graphics card with 2 gigabytes of video memory GDDR5, an Intel 3930K 6-core processor with 3.2GHz base clock speed and a Samsung 840 EVO SSD with 530 megabytes per second of read speed and 130 megabytes of write speed.

The system has been also tested on a laptop with 16 gigabytes of Random Access Memory, an Intel HD Graphics 620 on-board graphics card, an Intel Core i7-7500U 2-core processor with 2.70 GHz base clock speed and a Samsung 960 EVO NVMe M.2 SSD 3200 megabytes per second of read speed and 1900 megabytes per second

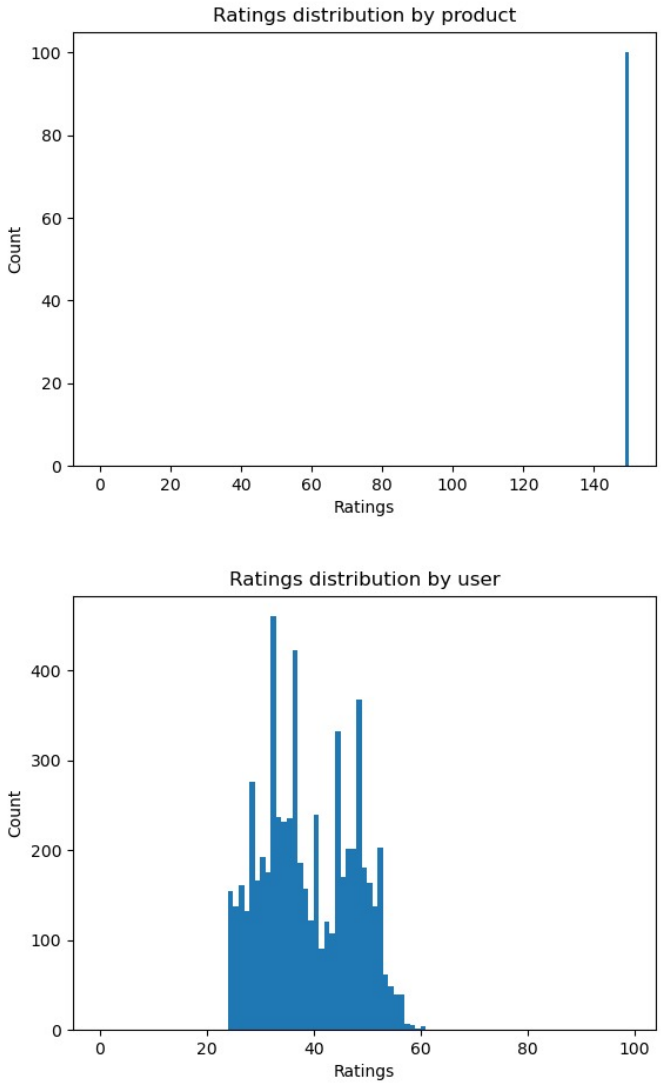

Figure 4.7: Jester 1 dataset distribution of train data.

of write speed.

4.4 Effectiveness and Explainability Evaluation

LastFM dataset results of the regressor are shown in Table 4.1; results are competing well with the rest of the methods and the FCP improves after the filtering. This filtering is useful to calculate the Model Fidelity of the test data, whose result was 0.3651. The Model Fidelity in that case is positive because one third of the test data was explainable. Regarding the Model Fidelity applied to a Top-N Recommender System, results are negative in Table 4.2. For this dataset, the proposed Recommender System, along with the exploration capabilities, would be recommended, as it can reach one third of accuracy while providing explanations. The reason for this affirmation is that, as the user explores the system and selects artists, it will improve the system and the explainability. It has also been considered that the dataset was

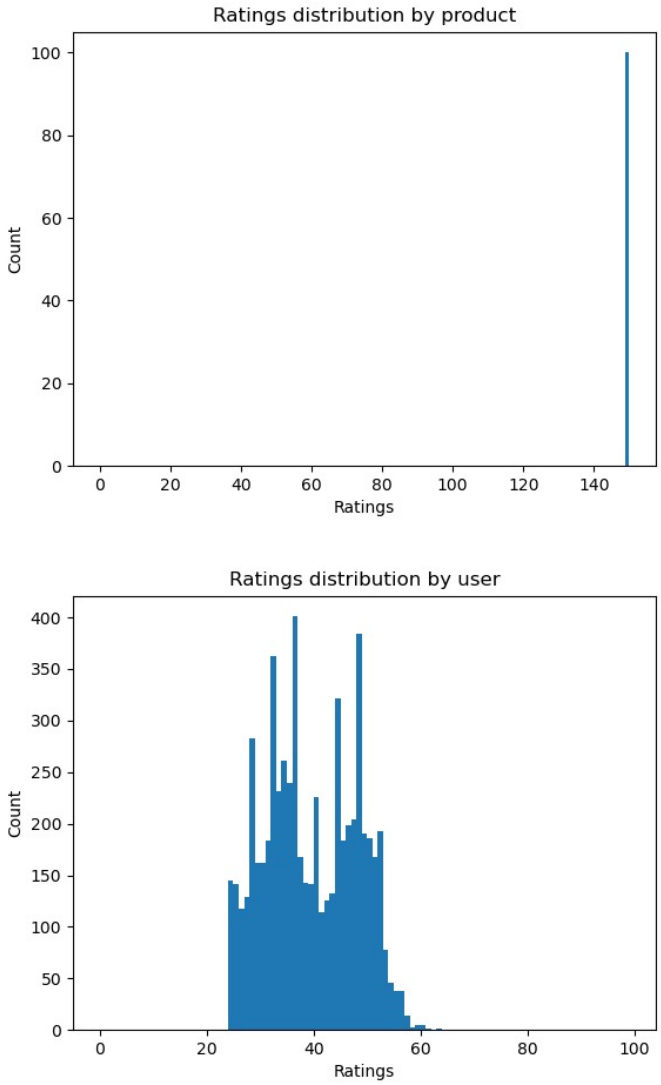

Figure 4.8: Jester 2 dataset distribution of train data.

not constructed with the user utilizing this Recommender System.

The Sushi dataset obtains a Model Fidelity value of 0.2103, which is lower than previous LastFM. Results from Table 4.3 show a strange repetition of the same FCP value, due to the fact that the dataset was fixed and all the survey participants were asked the same items; thus, FCP does not make sense here. Results of the regressor are moderately positive, but they do not beat the best models because more data would be needed and, in this type of fixed datasets, matrix is a better representation than a graph. The Model Fidelity indicates that one fifth of the recommendations would be accepted by the user and would be explainable, but again the user has the ability to explore and improve the system. The best results here are given by the Table 4.4, which shows, that with Top-N Recommender System and with these explanation techniques, the 70% of the recommendations can be explained. Recalling that Top-N sorts recommendation by the expected rating, the majority of them would not only obtain a positive rating, but they can be explained as well.

Method	MAE	RMSE	FCP
contextual	0.00213	0.01057	0.52423
contextual-filtered	0.00248	0.01081	0.53969
SVD	0.03036	0.05312	0.39660
SVDpp	0.01292	0.02339	0.41311
KMeans	0.00286	0.01096	0.54039
KNNBaseline	0.00259	0.01054	0.54408
KNNWithZScore	0.00256	0.01146	0.56616
CoClustering	0.00251	0.01073	0.00000

Table 4.1: Regressor results for the LastFM dataset.

N	Model Fidelity
2	0.07710
4	0.07428
6	0.07728
8	0.07850
14	0.07846
20	0.08050

Table 4.2: LastFM dataset Model Fidelity with Top-N recommendations provided by the Rating Regressor.

Method	MAE	RMSE	FCP
contextual	0.28610	0.32769	0.50000
contextual-filtered	0.28908	0.32953	0.50000
SVD	0.23274	0.28776	0.50000
SVDpp	0.22914	0.28439	0.50000
KMeans	0.23002	0.28787	0.50000
KNNBaseline	0.23051	0.28467	0.50000
KNNWithZScore	0.23004	0.28855	0.50000
CoClustering	0.38425	0.46207	0.50000

Table 4.3: Regressor results for the Shushi dataset.

N	Model Fidelity
2	0.70204
4	0.70634
6	0.71688
8	0.70139
14	0.53955
20	0.44991

Table 4.4: Sushi dataset Model Fidelity with Top-N recommendations provided by the Rating Regressor.

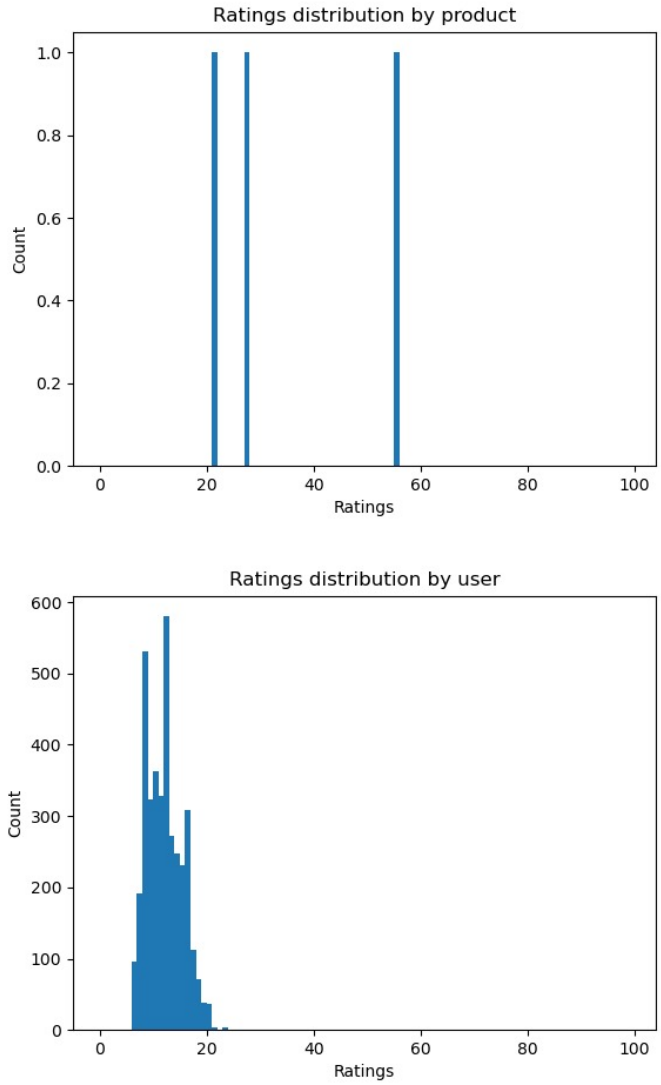

Figure 4.9: Jester 3 dataset distribution of train data.

Movie Tweetings dataset obtains a Model Fidelity of 0.2372, which means that nearly one fourth of recommendations would be explainable and would meet user preferences. Table 4.5 shows that FCP of proposed methods is unfavorable, as it cannot order well any of the regressed items because the regressions are not accurate at all. The error of the regressor is low and is close to the best results but does not outperform any of the best methods. The Table of Top-N recommendations and Model Fidelity of this model is the 4.6 and they do not show good results either. It could be thought that the proposed Recommender System would not work with this dataset, though in reality, more contexts could be added and also one out of four good and explainable recommendations is a result that can be kept using, while the model becomes better and better.

In the MovieLens 20K dataset a good Model Fidelity of 0.4916 is obtained with the test data, which means that half of the test data could be explained. Regressor results of Table 4.7 get near the best ones without surpassing them and the FCP

Method	MAE	RMSE	FCP
contextual	0.17372	0.23953	0.00000
contextual-filtered	0.17002	0.23382	0.00000
SVD	0.15400	0.20049	0.66582
SVDpp	0.14815	0.19325	0.68240
KMeans	0.14781	0.19329	0.68522
KNNBaseline	0.14562	0.19044	0.68908
KNNWithZScore	0.14688	0.19318	0.68382
CoClustering	0.52161	0.56703	0.55457

Table 4.5: Regressor results for the Movie Tweetings dataset.

N	Model Fidelity
2	0.13040
4	0.14419
6	0.14183
8	0.14158
14	0.14158
20	0.14083

Table 4.6: Movie Tweetings dataset Model Fidelity with Top-N recommendations provided by the Rating Regressor.

again gets stuck in 0 because of the lack of ordering power. Results from the Model Fidelity applied to Top-N of Table 4.8 mean that this Recommender System performs better with the proposed recommendation exploration rather than with the Top-N recommendations.

Results for MovieLens 20M, the largest dataset used, are unexpectedly positive with a Model Fidelity of 0.7987 for the proposed recommender system. The regressor obtains again close results in Table 4.9 without improving the best ones and a 0 in FCP. With Top-N recommendation Table 4.10 it is proved that they do not behave well in this dataset. These results show that, in this dataset, the best idea is to let the user freely explore the recommendation space as it will obtain positive and

Method	MAE	RMSE	FCP
contextual	0.16920	0.21907	0.00000
contextual-filtered	0.16838	0.21719	0.00000
SVD	0.13993	0.18290	0.69262
SVDpp	0.13622	0.17858	0.70331
KMeans	0.13626	0.17843	0.70822
KNNBaseline	0.13377	0.17581	0.71142
KNNWithZScore	0.13521	0.17805	0.70802
CoClustering	0.56994	0.60506	0.57032

Table 4.7: Regressor results for MovieLens 20K dataset.

N	Model Fidelity
2	0.08982
4	0.10393
6	0.10864
8	0.10875
14	0.11334
20	0.11454

Table 4.8: MovieLens 20K dataset Model Fidelity with Top-N recommendations provided by the Rating Regressor.

Method	MAE	RMSE	FCP
contextual	0.18024	0.24353	0.00000
contextual-filtered	0.17405	0.23290	0.00000
SVD	0.15344	0.19826	0.65918
SVDpp	0.14657	0.19038	0.68410
KMeans	0.14806	0.19253	0.68388
KNNBaseline	0.14560	0.18914	0.68755
KNNWithZScore	0.14706	0.19217	0.68425
CoClustering	0.53018	0.57415	0.56230

Table 4.9: Regressor results for MovieLens 20M dataset.

explainable recommendations.

In the other largest dataset, that is, Books Crossing, as happens in MovieLens 20M, testing provides good results for the proposed method with a Model Fidelity value of 0.7154, which is large. The regressor obtains nearly good results again and, at least, FCP values are positive, as shown in Table 4.11. Top-N Table 4.12 measures of explainability provide worse results discarding the possibility of using this method. In this dataset, a higher expected rating means less interpretability which can be indicating that contextual information is needed to make recommendations because people want similar books (books of the same collection, books of the same author or books of the same genre).

N	Model Fidelity
2	0.05170
4	0.05402
6	0.05319
8	0.05336
14	0.05672
20	0.06084

Table 4.10: MovieLens 20M dataset Model Fidelity with Top-N recommendations provided by the Rating Regressor.

Method	MAE	RMSE	FCP
contextual	0.30464	0.41169	0.50525
contextual-filtered	0.29666	0.40684	0.50048
SVD	0.28112	0.35072	0.51195
SVDpp	0.27632	0.34455	0.52585
KMeans	0.27254	0.34980	0.52872
KNNBaseline	0.28392	0.35060	0.53128
KNNWithZScore	0.27087	0.35119	0.53565
CoClustering	0.26788	0.42236	0.49484

Table 4.11: Regressor results for the Books Crossing dataset.

N	Model Fidelity
2	0.00054
4	0.00084
6	0.00084
8	0.00086
14	0.00082
20	0.00076

Table 4.12: Books Crossing dataset Model Fidelity with Top-N recommendations provided by the Rating Regressor.

Jester 1 dataset provides bad results in explainability with the full implementation of the Recommender System giving a Model Fidelity of 0.1423; while Top-N explainability results of Table 4.14 are very good. Results of the FCP get worse after the filtering in Table 4.13 and the regression is again nearly good but not the best.

Jester 2 improves the Model Fidelity to 0.198 as the results from the Top-N Table 4.16 worsen while regressor behaves well in Table 4.15.

The last Jester 3 dataset again improves previous Model Fidelity up to 0.2318, the results of the regressor of Table 4.17 being positive and Top-N results from Table

Method	MAE	RMSE	FCP
contextual	0.21763	0.26457	0.46141
contextual-filtered	0.21770	0.26461	0.45934
SVD	0.17489	0.21987	0.61479
SVDpp	0.17444	0.21966	0.61621
KMeans	0.17015	0.21438	0.63013
KNNBaseline	0.16935	0.21362	0.63098
KNNWithZScore	0.16790	0.21449	0.62986
CoClustering	0.54179	0.60194	0.00000

Table 4.13: Regressor results for the Jester 1 dataset.

N	Model Fidelity
2	0.69209
4	0.62453
6	0.60507
8	0.59625
14	0.59118
20	0.58077

Table 4.14: Jester 1 dataset Model Fidelity with Top-N recommendations provided by the Rating Regressor.

Method	MAE	RMSE	FCP
contextual	0.22173	0.26928	0.00000
contextual-filtered	0.22053	0.26770	0.00000
SVD	0.17584	0.22181	0.61060
SVDpp	0.17538	0.22157	0.61198
KMeans	0.17094	0.21642	0.62484
KNNBaseline	0.17021	0.21575	0.62564
KNNWithZScore	0.16830	0.21635	0.62369
CoClustering	0.53564	0.59831	0.00000

Table 4.15: Regressor results for the Jester 2 dataset.

N	Model Fidelity
2	0.44314
4	0.41667
6	0.42380
8	0.41247
14	0.38469
20	0.36971

Table 4.16: Jester 2 dataset Model Fidelity with Top-N recommendations provided by the Rating Regressor.

Method	MAE	RMSE	FCP
contextual	0.23027	0.27507	0.54515
contextual-filtered	0.23127	0.27664	0.54268
SVD	0.18977	0.23375	0.58273
SVDpp	0.18575	0.23058	0.58669
KMeans	0.18498	0.23254	0.58461
KNNBaseline	0.18896	0.23301	0.58298
KNNWithZScore	0.18308	0.23275	0.58417
CoClustering	0.51880	0.58630	0.00000

Table 4.17: Regressor results for the Jester 3 dataset.

N	Model Fidelity
2	0.29219
4	0.38239
6	0.39770
8	0.40883
14	0.38051
20	0.40187

Table 4.18: Jester 3 dataset Model Fidelity with Top-N recommendations provided by the Rating Regressor.

4.17 getting worse.

4.5 Summary

Surprisingly, with the little information provided that did not consider the real rating on the graphs and that embeddings also remove information in some way, the regressor performs very well and it could be enhanced by including information of the rating values in the graphs. The aforementioned confirms the first hypothesis.

Filtering the products that the user would not buy according to the proposed Recommender System does not improve the regressor but it does provide a large difference. That could mean that the second hypothesis does not get validated. This can also be positive due to the fact that the ANN is generalizing well.

There is a strange equilibrium between full implementation of proposed recommender and explainer and the Top-N recommender with the explainer proposed in this master thesis. Both provide opposite results in all tests. When the Top-N recommender provides high Model Fidelity, the other does not, and the contrary case also happens repeatedly. The most striking case is the one from Jester in which both metrics behave in this way; from Jester 1 to Jester 3 Model Fidelity of the full implementation increase while the Model Fidelity of the Top-N decrease. This leads to interpret that when third hypothesis is true, the fourth it is not and vice versa as it depends on the dataset.

This can lead to think that explainability is surely obtained always (if not with one recommender method, with the other), but there is always some explainability that can be captured for both methods. This can be because there is still more contextual information to be encoded, explored and exploited. Even though both Model Fidelity results do not sum 1 (and they do not have to) there is a loss in explainability, on which there can be a hint for improvement.

Very big datasets have shown to work better with the full proposed implementation, probably because there are more and more relationships that can be covered, and rules will be more powerful. Another reason can be that those datasets following a real power law are more likely to work with this method, since rules capture generic recommendations and MST obtains the backbone of the network. User will freely decide how to explore it; yet, it is very likely for all humans to have read certain books or watched certain movies simply because they are popular.

Chapter 5

Conclusions and Future Work

5.1 Conclusions

A recommendation problem can have many contextual information that can be interpreted in many ways. The system proposed in this master thesis provides every type of contextual information to be exploited in different ways, which allows explanation. The number of ways in which the contextual information can be exploited will depend on the computational power available and the type of explanations wanted.

With simple context aggregation rules, randomization and user input, the system obtains good recommendation results comparable to other well-known Recommender Systems. The good recommendation-explanation capabilities derived occur because of the implicit codification of relationships that other methods do not consider. In other words, this method eliminates successfully typical model assumption barriers by decomposing the context in different perspectives and grouping them.

In order to measure the accuracy of a recommendation, a regression is done, which is surprisingly capable to accurately regress with binary information-like weights of the user and product graph and its embeddings.

In addition to all mentioned above, the system is capable to explain in very simple natural language the recommendation with visual resources. These visual resources include community detection graphs and Maximum Spanning Trees of the graphs. The explanation of recommendation considers the necessary privacy concerns that are beneficial for both the user and the company. If the recommendation is given by more than one context, the explanation will be richer.

User preferences are considered with the implicit ratings that are already provided by the dataset. These preferences are also established with a group of parameters that let the user receive a variety of recommendation styles. These preferences are proved to be the ones that users have.

Results also show that this modular explainable Recommender System can be adapted to any recommendation selection algorithm (like Top-N), improving in some cases the explainability of the system.

5.2 Future Work

After all reviewing all the results three main improvement lines have been discovered that are expected to improve the proposed Recommender System in this master thesis. These improvement lines can be developed in separated forks or combined depending on the needs and research capacity. Contextual part of the system can be enhanced by adding more methods and this will improve explainability; explainability part can be enhanced by trying to understand in deeper way the context or some Deep Learning or Machine Learning methods could be used; the architecture could be adapted for production-ready systems while reducing computational time.

5.2.1 Context

The presented context understanding algorithms have been proved to provide useful information that is required to explain recommendations. As contexts are added to the system, the same product recommendations will contain as many explanations as context recommendations, with the target item matched in the user sampling. There are more algorithms that can also add explanations in similar ways.

In regard to the Community Detection, it would be interesting to try other Community Detection algorithms, since they can provide different assignments for the same product. These different assignments are not wrong at all because they provide a different perspective and thus they are highly acceptable. Two algorithms that would be relevant to add are the Multi-Level Community Detection algorithm [47], which tries to optimize modularity, and the Eigenvector Community Detection algorithm [48], which also tries to optimize modularity with the eigen decomposition of the graph matrix.

In this master thesis, the most simple rule-based association discovery algorithm is used, which is Apriori. This algorithm was slightly inefficient because it lasted long to compute the rules and, sometimes, after starting the system to run with a minimum confidence value, it did not find any rule owing to that value. This parameterization is tricky because, if the minimum confidence is very high, it will notfind any rule and, if the minimum confidence is very low, it will last too long computing the rules. It would be a great improvement to avoid that minimum confidence parameter and there are two association discovery algorithms (ASSOC and OPUS search) that do not need it. There are other families of association discovery that need other types of parameters, which can be very useful. An example of another family is the K-optimal Rule Mining, which only needs the number of rules that are wanted. With this parameter, the part of the parameter search would not be necessary, and it would be easier to detect how many recommendations would the system generate per each user.

Detecting which contexts share large similarities and thus are redundant could also improve the explainability capabilities of the system. It would be interesting to develop a system that automatically computes as many different contexts as possible in a limited time. This automation could be done with intelligent reinforcement signals or with a hand-crafted algorithm that would try to find Pareto efficiency (maximize explainability with number of methods and minimize time). Then, contexts could be understood as features, and a feature selection/optimization could be performed with the metrics of the contexts and the recommendations provided. In this way, similar contexts or contexts that do not help enhance the results of explainability measures proposed could be discarded or even replaced.

As an interesting feature, making use of the embeddings, the user could also receive information about it and about the items provided in the recommendation. Performing the cosine distance between the embedding of the user, its friends and some people close to it and showing it with a simple algorithm (t-SNE) could let the user see which friends are closer to it and could also see that the user "is not alone" and there are people with very similar tastes. Naturally, friends' names would be revealed, but not the names of other people. For the items provided in the recommendation, it would be interesting to show the items that are already bought by the user, some similar ones that do not appear in the recommendation and the ones that appear in the recommendation. In this manner, the user could see how the items are distributed in the similarity space.

5.2.2 Explainability

Contexts usually depend on other contexts. We could understand that the context understanding is a recursive task. For instance, in order to understand a news article, all the information contained depends on other news articles and information that can be found in Wikipedia, but a Wikipedia article is also linked to other articles and web resources.

Finding a way to match all those contexts and provide the explanations recursively and in a simplified way to understand could be an interesting idea. That context recursive look-up could be done randomly, by trying to optimize the depth of the explanation or giving the user the possibility to limit that depth.

There should also be a way to provide abstraction on how the explanations are created. With a language generative model that could explain the relationships in a deep, varied and rich way, the explanation could be improved. An idea for attaining that could be the creation of a simple Mechanical Turk Task where users are provided with a linked structure and where an explanation of links from one point to the other has to be given by the users. Then, the links could be changed (by the product name, for example), so that the model could learn to traverse the nodes and understand the underlying information. Finally, the explanations of the model could be enhanced to join different explanations in one.

The previously ideated task can be separated in two main objectives: finding the best explanatory path(s) and provide a reasonable explanation. If many paths are provided, the explanation should not be repetitive. Paths are not simple 1D paths, but navigation among contexts. This could be achieved by some reinforcement learning agents where the reward function could be modulated depending on the desired short path and on the desired different contexts, providing the embeddings of every node, which is traversed as information.

All this information could be stored as valid examples to generate a unique model that would integrate the traversion of nodes and the explanation generation. In that approach, the most basic data are inputs and explanations are outputs. All the contexts would be latent in the Deep Learning model. Another solution to this task could be implementing a Deep Learning transformer that could learn how to encode the context and decode the explanation.

5.2.3 Architecture

The architecture proposed in this thesis works for many datasets and provides positive results. Notwithstanding, as mentioned in the architecture definition, there are some imperfections that could be addressed in future versions. Figure 5.1 shows an architecture proposal for the future version of that recommender. Some operations will be adapted to use multiprocessing features whenever it is possible.

The main disadvantages with respect to other Recommender Systems are the initial computations speed for calculating all the graphs and rules, as well as the big amount of computing resources it needs. In order to correct the initial computations speed, the next section provides an interesting analysis of the current architecture and a new one is proposed, so that the speed will be increased, the context inclusions are easier and the computational resources are used more efficiently.

This proposed architecture would have the ability to load the dataset also from the production database. Then, the pipeline would be divided into two possible forks: development and production. The development step would behave as the previous architecture with one exception on the computations: the production part would update the database instead of regenerating it. For an enhanced performance, the data storage would also include a Graph database with the features of performing Community Detection algorithms and Maximum Spanning Trees; that database could be Neo4J. The document storage would include a script to perform any Rule Mining algorithm and the proposed database, as in the previous architecture, MongoDB.

This architecture proposal would allow online deployment and updates at any time without affecting the service.

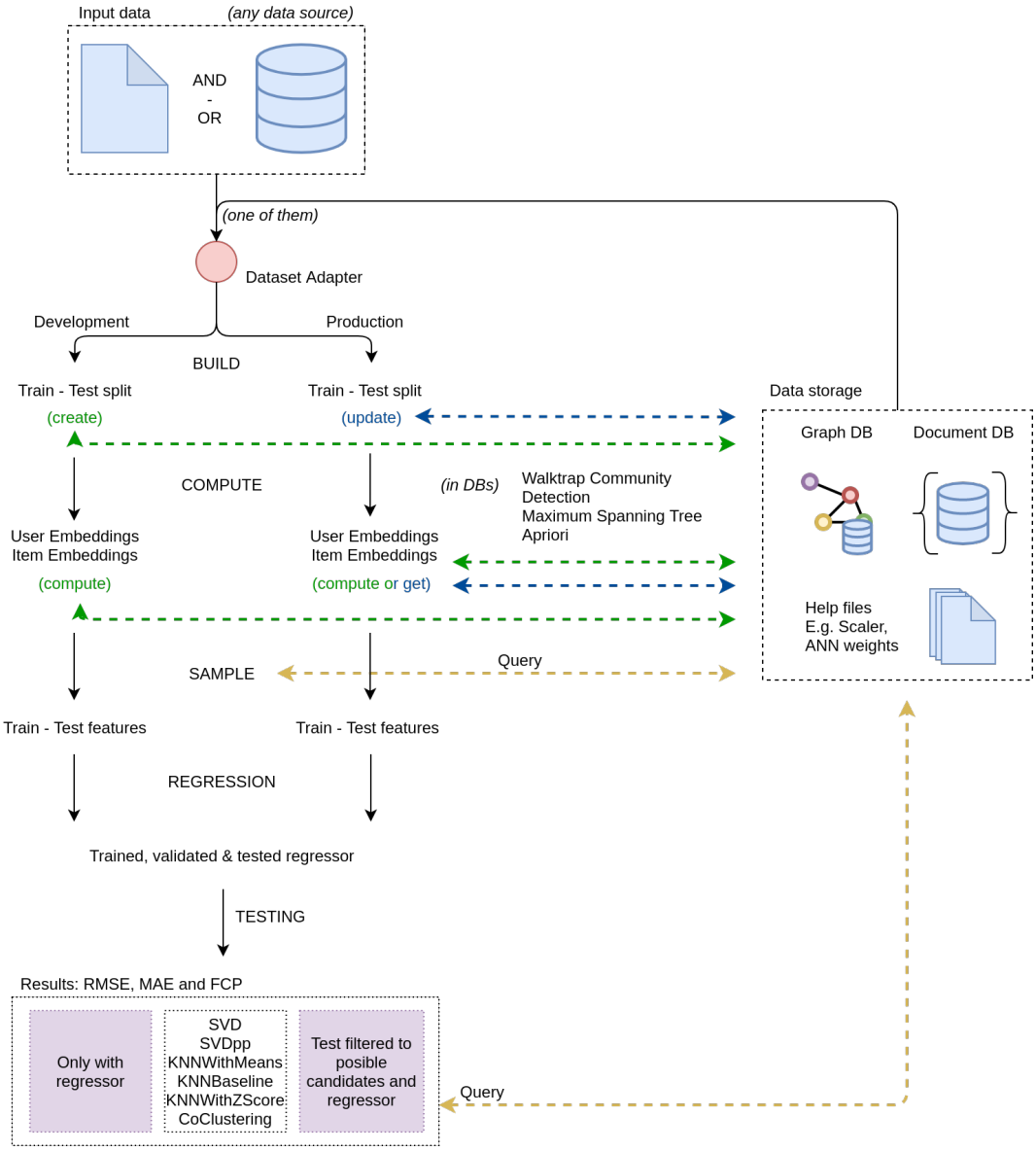

Figure 5.1: Future architecture specification.

References

[1] Unai Zulaika, Asier Gutiérrez, and Diego López-de Ipiña. Enhancing profile and context aware relevant food search through knowledge graphs. *Proceedings*, 2(19):1228, Oct 2018.

[2] Moshe Unger. Latent context-aware recommender systems. In *Proceedings of the 9th ACM Conference on Recommender Systems*, RecSys '15, page 383–386, New York, NY, USA, 2015. Association for Computing Machinery.

[3] Yongfeng Zhang and Xu Chen. Explainable recommendation: A survey and new perspectives. *Foundations and Trends® in Information Retrieval*, 14(1):1–101, 2020.

[4] Yichao Lu, Ruihai Dong, and Barry Smyth. Why i like it: Multi-task learning for recommendation and explanation. In *Proceedings of the 12th ACM Conference on Recommender Systems*, RecSys '18, page 4–12, New York, NY, USA, 2018. Association for Computing Machinery.

[5] Kosetsu Tsukuda and Masataka Goto. Dualdiv: Diversifying items and explanation styles in explainable hybrid recommendation. In *Proceedings of the 13th ACM Conference on Recommender Systems*, RecSys '19, page 398–402, New York, NY, USA, 2019. Association for Computing Machinery.

[6] Yehuda Koren and Joseph Sill. Collaborative filtering on ordinal user feedback. In *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence*, IJCAI '13, page 3022–3026. AAAI Press, 2013.

[7] Konstantina Christakopoulou and Arindam Banerjee. Adversarial attacks on an oblivious recommender. In *Proceedings of the 13th ACM Conference on Recommender Systems*, RecSys '19, page 322–330, New York, NY, USA, 2019. Association for Computing Machinery.

[8] Huafeng Liu, Jingxuan Wen, Liping Jing, and Jian Yu. Deep generative ranking for personalized recommendation. In *Proceedings of the 13th ACM Conference on Recommender Systems*, RecSys '19, page 34–42, New York, NY, USA, 2019. Association for Computing Machinery.

[9] Maurizio Ferrari Dacrema, Paolo Cremonesi, and Dietmar Jannach. Are we really making much progress? a worrying analysis of recent neural recommendation approaches. In *Proceedings of the 13th ACM Conference on Recommender Systems*, RecSys '19, page 101–109, New York, NY, USA, 2019. Association for Computing Machinery.

[10] Zhu Sun, Jie Yang, Jie Zhang, Alessandro Bozzon, Long-Kai Huang, and Chi Xu. Recurrent knowledge graph embedding for effective recommendation. In *Proceedings of the 12th ACM Conference on Recommender Systems*, RecSys '18, page 297–305, New York, NY, USA, 2018. Association for Computing Machinery.

[11] Enrico Palumbo, Giuseppe Rizzo, and Raphaël Troncy. Entity2rec: Learning user-item relatedness from knowledge graphs for top-n item recommendation. In *Proceedings of the Eleventh ACM Conference on Recommender Systems*, RecSys '17, page 32–36, New York, NY, USA, 2017. Association for Computing Machinery.

[12] Tianshu Lyu, Fei Sun, Peng Jiang, Wenwu Ou, and Yan Zhang. Compositional network embedding for link prediction. In *Proceedings of the 13th ACM Conference on Recommender Systems*, RecSys '19, page 388–392, New York, NY, USA, 2019. Association for Computing Machinery.

[13] Ahmed Rashed, Josif Grabocka, and Lars Schmidt-Thieme. Attribute-aware non-linear co-embeddings of graph features. In *Proceedings of the 13th ACM Conference on Recommender Systems*, RecSys '19, page 314–321, New York, NY, USA, 2019. Association for Computing Machinery.

[14] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[15] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. Albert: A lite bert for self-supervised learning of language representations, 2019.

[16] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach, 2019.

[17] Mingxing Tan and Quoc V. Le. Efficientnet: Rethinking model scaling for convolutional neural networks, 2019.

[18] HeeJae Jun, ByungSoo Ko, Youngjoon Kim, Insik Kim, and Jongtack Kim. Combination of multiple global descriptors for image retrieval, 2019.

[19] Adrian Bulat, Jean Kossaifi, Georgios Tzimiropoulos, and Maja Pantic. Toward fast and accurate human pose estimation via soft-gated skip connections, 2020.

[20] Z. Wang, X. Ye, C. Wang, J. Cui, and P. Yu. Network embedding with completely-imbalanced labels. *IEEE Transactions on Knowledge and Data Engineering*, pages 1–1, 2020.

[21] Qi Zhu, Hao Wei, Bunyamin Sisman, Da Zheng, Christos Faloutsos, Xin Luna Dong, and Jiawei Han. Collective multi-type entity alignment between knowledge graphs.

[22] Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination, 2019.

[23] Stephen Bonner and Flavian Vasile. Causal embeddings for recommendation. In *Proceedings of the 12th ACM Conference on Recommender Systems*, RecSys '18, page 104–112, New York, NY, USA, 2018. Association for Computing Machinery.

[24] Rahul Makhijani, Shreya Chakrabarti, Dale Struble, and Yi Liu. Lore: A large-scale offer recommendation engine with eligibility and capacity constraints. In *Proceedings of the 13th ACM Conference on Recommender Systems*, RecSys '19, page 160–168, New York, NY, USA, 2019. Association for Computing Machinery.

[25] Marie Al-Ghossein, Pierre-Alexandre Murena, Talel Abdessalem, Anthony Barré, and Antoine Cornuéjols. Adaptive collaborative topic modeling for online recommendation. In *Proceedings of the 12th ACM Conference on Recommender Systems*, RecSys '18, page 338–346, New York, NY, USA, 2018. Association for Computing Machinery.

[26] Jaron Harambam, Dimitrios Bountouridis, Mykola Makhortykh, and Joris van Hoboken. Designing for the better by taking users into account: A qualitative evaluation of user control mechanisms in (news) recommender systems. In *Proceedings of the 13th ACM Conference on Recommender Systems*, RecSys '19, page 69–77, New York, NY, USA, 2019. Association for Computing Machinery.

[27] Amy A. Winecoff, Florin Brasoveanu, Bryce Casavant, Pearce Washabaugh, and Matthew Graham. Users in the loop: A psychologically-informed approach to similar item retrieval. In *Proceedings of the 13th ACM Conference on Recommender Systems*, RecSys '19, page 52–59, New York, NY, USA, 2019. Association for Computing Machinery.

[28] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'14, page 2672–2680, Cambridge, MA, USA, 2014. MIT Press.

[29] Joseph B. Kruskal. On the shortest spanning subtree of a graph and the traveling salesman problem. *Proceedings of the American Mathematical Society*, 7(1):48–50, 1956.

[30] Meichen Yu, Arjan Hillebrand, Prejaas Tewarie, Jil Meier, Bob van Dijk, Piet Van Mieghem, and Cornelis Jan Stam. Hierarchical clustering in minimum spanning trees. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 25(2):023107, 2015.

[31] Pascal Pons and Matthieu Latapy. Computing communities in large networks using random walks. In *Proceedings of the 20th International Conference on Computer and Information Sciences*, ISCIS'05, page 284–293, Berlin, Heidelberg, 2005. Springer-Verlag.

[32] Manpreet Kaur and Shivani Kang. Market basket analysis: Identify the changing trends of market data using association rule mining. *Procedia Computer Science*, 85:78 – 85, 2016. International Conference on Computational Modelling and Security (CMS 2016).

[33] Yusuf Kurnia, Yohanes Isharianto, Yo Ceng Giap, Aditiya Hermawan, and Riki. Study of application of data mining market basket analysis for knowing sales pattern (association of items) at the o! fish restaurant using apriori algorithm. *Journal of Physics: Conference Series*, 1175:012047, mar 2019.

[34] Herman Aguinis, Lura E. Forcum, and Harry Joo. Using market basket analysis in management research. *Journal of Management*, 39(7):1799–1824, 2013.

[35] Rakesh Agrawal and Ramakrishnan Srikant. Fast algorithms for mining association rules in large databases. In *Proceedings of the 20th International Conference on Very Large Data Bases*, VLDB '94, page 487–499, San Francisco, CA, USA, 1994. Morgan Kaufmann Publishers Inc.

[36] Claire Donnat, Marinka Zitnik, David Hallac, and Jure Leskovec. Learning structural node embeddings via diffusion wavelets. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '18, page 1320–1329, New York, NY, USA, 2018. Association for Computing Machinery.

[37] Aditya Grover and Jure Leskovec. Node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, page 855–864, New York, NY, USA, 2016. Association for Computing Machinery.

[38] Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 632–642, Lisbon, Portugal, September 2015. Association for Computational Linguistics.

[39] Silvia Milano, Mariarosaria Taddeo, and Luciano Floridi. Recommender systems and their ethical challenges, 04 2019.

[40] T. George and S. Merugu. A scalable collaborative filtering framework based on co-clustering. In *Fifth IEEE International Conference on Data Mining (ICDM'05)*, pages 4 pp.–, 2005.

[41] Nicolas Hug. Surprise, a Python library for recommender systems. http://surpriselib.com, 2017.

[42] Georgina Peake and Jun Wang. Explanation mining: Post hoc interpretability of latent factor models for recommendation systems. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '18, page 2060–2069, New York, NY, USA, 2018. Association for Computing Machinery.

[43] T. Kamishima and S. Akaho. Efficient clustering for orders. In *Sixth IEEE International Conference on Data Mining - Workshops (ICDMW'06)*, pages 274–278, 2006.

[44] Simon Dooms, Toon De Pessemier, and Luc Martens. Movietweetings: a movie rating dataset collected from twitter. In *Workshop on Crowdsourcing and Human Computation for Recommender Systems, CrowdRec at RecSys 2013*, 2013.

[45] Cai-Nicolas Ziegler, Sean M. McNee, Joseph A. Konstan, and Georg Lausen. Improving recommendation lists through topic diversification. In *Proceedings of the 14th International Conference on World Wide Web*, WWW '05, page 22–32, New York, NY, USA, 2005. Association for Computing Machinery.

[46] Kenneth Goldberg, Theresa Roeder, Dhruv Gupta, and Chris Perkins. Eigentaste: A constant time collaborative filtering algorithm. *Information Retrieval*, 4:133–151, 07 2001.

[47] Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2008(10):P10008, Oct 2008.

[48] M. E. J. Newman. Finding community structure in networks using the eigenvectors of matrices. *Physical Review E*, 74(3), Sep 2006.