Faculty of Sciences Departement of Computer Science

MASTER'S THESIS

Unveiling Transparency and Trust in Recommender Systems through Model-Agnostic Counterfactual Explanations

Author : GAILLY Vincent 547819

Promotor : Mr. SACHARIDIS Dimitris

Academic year 2023–2024

Contents

List of Figures

List of Tables

Acknowledgements

Before starting my thesis, I would like to express my deep gratitude to all the people who contributed to the completion of this work.

First and foremost, I wish to express my gratitude to my supervisor, Mr. Dimitris Sacharidis, whose expertise and guidance have greatly enriched this thesis. I would also like to thank him sincerely for his availability to answer any questions I may have had.

I would also like to thank my family for uplifting me, supporting me, and providing valuable advice throughout my life and studies.

Lastly, I would like to thank my classmates Thomas, Oruç, and Maxime, with whom I have collaborated on numerous projects during this master's program and without whom, these years would not have gone so fast.

Abstract

Artificial intelligence (AI), particularly machine learning (ML), has significantly advanced, offering vast prospects for technological innovation. However, its deployment in critical areas such as healthcare, justice, and finance raises ethical and trust issues due to its ”black-box” nature. In high-risk environments, understanding the reasoning behind AI predictions is essential, leading to the necessity of explainability in AI. This concept aims to make AI models’ decision-making processes transparent and understandable.

In recommender systems (RS), explainability is crucial due to the complexity of these systems developed to provide personalized recommendations. This thesis focuses on model-agnostic counterfactual explanations, which create hypothetical scenarios to observe changes in model outcomes. These explanations are universal, treating the model as a black box, and do not rely on a proxy.

The primary objective of this thesis is to extend existing work to address ”why not questions” in RS. These questions inquire why a specific item is not recommended to the user. This new scenario introduces a significant challenge: the search space for finding counterfactual explanations becomes exponentially large. Therefore, it was necessary to introduce sampling techniques to reduce the search space size.

Two techniques were developed: one using a Jaccard similarity matrix and another leveraging the recommendation system itself to find similar items to the target. While promising, these methods have limitations, such as privacy issues and cold start problems.

The thesis also conducted tests to ensure the proper functioning of search strategies and replicated previous experiments to compare heuristic candidate search strategies. The results were consistent across various strategies.

Future work includes leveraging differences and similarities in scenarios, introducing adaptive sampling in the ″why not″ scenario, and applying sampling techniques to the ″why″ scenario. This research provides valuable insights into adapting model-agnostic explanation mechanisms to answer ″why not questions″ in recommendation systems, inspiring further investigation and innovation in the field.

Glossary

AI Artificial Intelligence

BFS Breadth First Search

CAMs Class Activation Maps

CAVs Concept Activation Vectors

CBF Content-Based Filtering

CF Collaborative Filtering

CNNs Convolutional Neural Networks

GAMs Global Attribution Mappings

k-NN K-Nearest Neighbors

LIME Local Interpretable Model-Agnostic Explanations

LRP Layer-wise Relevance Propagation

ML Machine Learning

RS Recommender System

SHAP SHapley Additive exPlanation

XAI Explainable Artificial Intelligence

Chapter 1

Introduction

1.1 Motivation

Artificial intelligence (AI) and more precisely the machine learning (ML), has seen significant advances in recent years, opening up vast prospects in terms of technological innovation and automation in various work situations. However, this deployment of AI systems is also affecting more critical areas such as healthcare, justice and finance, introducing ethical and trust issues as a result of the difficulty of understanding their inner workings (″black-box″ nature).

In certain scenarios, and particularly in high-risk environments where AI systems produce recommendations or predictions that influence our choices, obtaining the model's prediction alone (the ″what″) may not be enough to fully solve the problem in question. It therefore becomes necessary for the model to provide an explanation of the reasoning behind its prediction (the ″why″) to enable users to make an informed decision. However, this information appeals not only to end-users but also to AI developers, who require assurance that the technology functions according to their specifications. It's equally significant for experts who want to ensure that the model acts ethically, fairly and in accordance with human values. This need has given rise to the concept of explainability in AI, an important area of research that aims to make the inner workings of complex AI models more transparent and provide understandable information about their decision-making processes.

In the field of recommender systems (RS), explainability is of unique importance. RS is a sub-field of AI that arose as a result of the significant development of e-commerce and its need to provide personalized recommendations to customers to enhance their user experience. Advancements in this field necessitate the utilization of ever more complex techniques to produce more refined recommendations. However, this progression often leads to the emergence of the "black box" phenomenon within RSs. The challenge now is to strike a balance between enhancing the quality of recommendations and providing interpretable explanations for these recommendations.

Naturally, there has been a great deal of research into "interpretable" RSs, which by their very nature are easy to explain and whose choice of recommendations can be justified. However, there are cases where this is not feasible, particularly when the model is optimized for performance and is therefore not intrinsically interpretable. Another situation occurs when the model cannot be modified, the RS is not accessible and the resulting recommendations are used as a service. In such situations, the recommendations must be explained post-hoc once the system has been deployed. To address this, a new model is used to explain the recommendations, which can be categorized into two distinct types: model-intrinsic and model-agnostic. Model-intrinsic techniques leverage specific characteristics inherent to a particular model to generate explanations, resulting generally in a higher level of precision. However, their applicability is limited to specific models. Conversely, the second approach, extensively explored in this thesis, treats the model as a black box, relying solely on input-output interactions to deliver explanations in a more universally applicable manner.

This thesis will specifically focus on a type of explanation for RSs : model-agnostic counterfactual explanation. Counterfactual explanations involve creating hypothetical scenarios in which small changes are made to input data to observe how these changes affect the model's outcomes. This approach offers the advantage of universality, treating the model as a black box, while maintaining the same fidelity and precision as a model-specific approach. Counterfactuals achieve the desired result within the RS whithout relying on a proxy, unlike most model-agnostic approaches.

As explained previously, to obtain counterfactual explanations, small changes are made to the model's input data, resulting in a change in the output. Each of these possible changes represents a candidate explanation, and several of them may produce the desired outcome. To find the best explanation, i.e., the most concise and efficient change in the input data, it requires exploring all possible combinations of changes. This makes the search space exponentially large, depending on the number of elements that can be modified.

In practice, a brute-force algorithm that explores all possible combinations is generally not feasible due to the time required to traverse the entire search space. For this reason, several heuristic strategies have been proposed and studied, as described in the paper entitled "Model-Agnostic Counterfactual Explanations for Recommendation Systems"[14], co-authored by my promotor, Mr. Sacharidis. In this article, where counterfactual explanations are used to explain why a certain recommendation was given by the RS to a user, the exploration of candidate explanations is done with a limited budget. The entire search space cannot therefore be visited, which is why these strategies are proposed : to start by exploring the most promising candidates.

1.2 Problem definition

The objective of this thesis is to extend the existing work to provide explanations for another type of inquiry known as "why not questions" in the literature. In this scenario, users wonder **why** a specific item is **not** in their recommendation list. To answer such a question, we decided to provide an explanation to the users, showing them which item(s) they should have interacted with to obtain the desired item in their recommendations. The idea now is no longer to find items to delete from the user's history, but rather items to add.

However, a new challenge arises in this scenario : the search space for finding a counterfactual explanation is generally much larger. In the previous scenario, the search space consisted of all the different subsets of the users' histories and was therefore exponentially larger than the size of the users' interaction lists. Here, the search space grows exponentially with the number of items the users have not interacted with, which can sometimes be enormous when you look at today's giant e-commerce platforms.

Consequently, adapting the algorithm which use the different strategies and devising new approaches are necessary to meet this new requirement, which is the focus of this thesis.

1.3 Contribution

To extend the work done in the referenced article and conduct my experiments within the same setup, I reused the code from the original GitHub repository associated with the article.

1.3.1 Code cleanup and documentation

The first step was to familiarize myself with this code. Given that the code was three years old, it presented several challenges : it contained unused segments, was limited in comprehensive documentation, and did not clearly outline the dependencies and installation steps required to run it. This lack of clarity made the code complex to understand and use.

Initially, a new GitHub[1] repository was created, and all content from the original repository was migrated. Subsequent modifications were introduced, beginning with a comprehensive review and revision of the Python code (identification and removal of redundant or unused portions of code) within the notebooks. Inclusion of detailed textual documentation was a primary focus, elucidating the conceptual underpinnings of each code segment representing a strategy and the approach to evaluate them.

Moreover, a comprehensive README file was crafted, outlining the structure of the GitHub repository. It provided comprehensive instructions on executing the code, installing necessary dependencies, and utilizing now a PipFile to automate dependency installation.

[1] `https://github.com/SnewZz/Thesis`

This concerted effort in enhancing the GitHub documentation and executing a systematic reorganization aimed to establish a more user-friendly and comprehensible platform. For more information on the new structure of the GitHub repository, please refer to the README file.

1.3.2 Extension of the code to the why not scenario

Initially, it was necessary to imagine a way to adapt the code to be able to reuse the strategies already coded in the algorithm that will provide an explanation for this new scenario. It was therefore necessary to recover the new search space in which they would operate: all possible subsets stemming from the list of non-interacted items.

This list being too large for the strategies to have time to find a solution with the allocated budget, a sampling step had to be carried out. Two methods have been developed, allowing to recover a number of items most similar to the target item: the first uses a jaccard similarity matrix of items to items and the second uses recommender systems to find the items it would advise if we had just asked the target item.

Finally, I also had to adapt the code to evaluate a candidate explanation, the selection criteria no longer being exactly the same: the target item should no longer approach the top recommendations but move away from them and the size of the user interaction list should no longer decrease but increase. The score assigned to each candidate was therefore adjusted accordingly.

1.3.3 Evaluation of the new code

I also coded two experiments to evaluate the performance and consistency of the code I produced. The first is to verify that the changes made do not disrupt the functioning of search space exploration strategies. The second aims to evaluate the performance of the two proposed sampling methods to reduce the size of the search space.

1.4 Outline

Chapter 2, "Background" provides the theoretical background necessary to understand the work undertaken in this thesis. It begins by defining recommender systems and classifying the various types, highlighting their differences. Subsequently, the chapter delves into the field of explainability in artificial intelligence, categorizing explainability methods and elucidating the types of transparent models. Finally, it examines explainability in recommender systems, outlining the different methods employed and their characteristics.

Chapter 3, ”Related work”, will present the article upon which my dissertation is based. It will offer a formal definition of the addressed problem, along with the approach and proposed solutions/strategies.

Chapter 4, ”Methodology”, will focus on the methodology used to extend explainability to the ”why not scenario.” This chapter will delve deeper into the problem addressed in the dissertation, discussing the main challenges encountered. Additionally, it will outline the approach employed to tackle this problem, providing an overview of the algorithm.

Chapter 5, ”Experimental evaluation”, will analyze the obtained results and draw observations from them. This chapter will also showcase the two different evaluations conducted to ensure the consistency of the experiments.

Chapter 6, ”Future works”, will provide possible avenues for extending this work. Some of tese ideas arose in the course of writing this thesis, but due to time constraints could not be realized.

Chapter 7, ”Conclusion”, will summarize the key findings and contributions of the thesis.

Chapter 2

Background

2.1 Recommender systems

2.1.1 Introduction

The need for recommendations arose in the mid-1990s with the development of e-commerce sites. These sites offered a wide range of items/services, providing greater freedom and autonomy for users. However, this abundance of choice also led to a problem: users often felt overwhelmed by the multitude of possibilities. The phrase "too much choice kills choice" became relevant, as an overabundance of options negatively impacts user experience. To address this issue, **recommender systems (RS)** were developed.

RS also known as recommendation engines, are algorithms designed to suggest relevant items to the users. In simple terms, these algorithms try to generate a ranked list of products or services that might be of most interest to the user. RSs are dynamic information processing systems that construct personalized recommendations by actively collecting various types of data in their database. This data can be broadly classified into three categories :

(a) **Items :** These are all the different products or services that the RS can suggest, such as movies, e-commerce products, news articles or other items related to the field of application. You can also find informations/features related to these items. The complexity of gathering this information and its relevance depend on the item itself. For instance, images require specific image processing algorithms to extract features, while text-based items such as news articles require algorithms that can extract relevant information, etc. These features are notably indispensable for content-based filtering, a common approach used in RSs, as they help match items to user preferences based on their content similarities (see Section 2.1.3 (Classification of recommender systems)).

(b) **Users :** This is all the data collected about users. This plays a key role in the quality of a RS, as each user has different goals, preferences and characteristics that influence the personalization process. In the literature, this collection of information is often referred to as the "user model", encapsulating the user's desires and needs. The selection of user information highly depends on the recommendation technique used. For example, context-aware RSs incorporate contextual information to tailor recommendations based on specific situations (e.g. in a weather context, a travel recommender will take into account what time of year it is, not suggesting the same destinations if it's winter or summer). In contrast, collaborative filtering represents the user by a list of ratings for specific items. So there are various effective ways to represent user data (see Section 2.1.3 for more details).

(c) **Interactions between users and items :** It refers to the recorded connections between users and items, such as ratings, likes/dislikes, clicks, views, etc. This data collected from the interactions is crucial for recommendation algorithms to provide personalized suggestions. Most of this data is user feedback, which can be either implicit or explicit. Explicit feedback is intentionally provided by the user, it can take the form of numerical evaluations, votes, ratings, etc. For instance, users can rate a restaurant with stars based on their dining experience. Implicit feedback is inferred from user behavior, like clicks or adding items to a shopping cart. This type of feedback is a unary rating and indicates a positive preference; there is no negative rating in this type of rating. Some recommendation techniques use this data to refine their algorithms, as in collaborative filtering, which identifies similar items or users based on these interactions.

RSs play an important role in filtering all the items a platform can offer. This technology is highly valued in various industries because, when used effectively, it helps companies stand out from competitors and increase revenue. Consequently, many companies invest in research in this burgeoning field and organize events with significant rewards. One notable example is the Netflix Prize, which ran from 2006 to 2009. The company offered a sum of 100 million dollars to the team that managed to deliver an RS 10% more accurate than their existing algorithm.

2.1.2 Applications and examples

RSs are now widely used in various fields, each tailored to specific domains. Below are some concrete examples of their applications:

(a) **E-commerce** : Amazon is a popular platform that utilizes RSs, particularly item-based collaborative filtering to personalize online shopping for each user. By leveraging user data such as search histories and previous purchases, Amazon suggests items that are most relevant to individual users, enhancing their shopping experience and enabling them to discover items they might not have found among the vast product catalog.

(b) **Social Media** : Social media platforms, like Facebook and Twitter, employ recommendation systems to curate users' feeds and recommend content, friends, groups, and advertisements. These systems aim to keep users engaged by showing them relevant and interesting content based on their interactions and preferences.

(c) **Media Streaming** : Services like Netflix, Spotify, and YouTube extensively use recommendation systems to suggest movies, TV shows, songs, and videos to their users. These platforms analyze users' viewing or listening histories, ratings, and preferences to provide personalized recommendations, increasing user satisfaction and retention.

2.1.3 Classification of recommender systems

In the literature, there are generally four basic approaches: **content-based**, **collaboration filtering**, **demographic** and **knowledge-based**.

However, since RSs now have a fairly long history, many different techniques have emerged that depart from the traditional approaches. There are also so-called "**hybrid**" approaches that combine several techniques in order to get the best out of each of them (this is notably the case of the winners of the Netflix prize, who use the ideas of many types of hybrids [11]), which increases the number of possibilities and makes classification difficult.

It wasn't until Burke's survey [7] that we had a formal categorization of these hybrid techniques. He subdivided the hybrid RSs into 7 categories: **Weighted**, **Switching**, **Mixed**, **Feature Combination**, **Feature Augmentation**, **Cascade**, **Meta-level**.

2.1.3.1 Collaborative filtering recommender system

The Collaborative Filtering (CF) approach was one of the first to emerge. To imagine it, the developers drew on a straighforward observation: people frequently depend on recommendations given by others when making their everyday decisions. The algorithm even takes the idea a step further, based on the assumption that users who had the same tastes/needs in the past are likely to be looking for the same thing in the future.

So CF doesn't need explicit knowledge about items or users, but rather the history of interactions between items and users in order to generate recommendations. Among CF RSs, one big choice to make which can influence the performance depending on the application :

(a) **User-based Collaborative filtering :** In this approach, the system looks for users with the same tastes as the user for whom it is making recommendations, and suggests items with which similar users have interacted. The similarity calculated by the system is based on techniques such as cosine similarity and pearson correlation. The advantage of this approach over the second is that it generally offers a better user experience, as it tends to make more original recommendations.

(b) **Item-based Collaborative filtering :** Here, the system tries instead to identify items in which the target user has shown an interest, and then suggests them in its recommendations. In this case, too, the RS needs the user's interactions in order to see which items he or she has previously interacted with. This approach is particularly computationally efficient for recommenders who have to operate in an environment with many users. It also needs to be updated less frequently, as the number of items varies less in this kind of application domain.

The CF approach has its advantages, such as providing personalized recommendations and not requiring explicit knowledge about items and users, but only implicit knowledge about their interactions. However, it also has disadvantages. One of the main issues is sparsity, where the system may lack sufficient data to make accurate recommendations. Another well-known problem is the "cold-start," meaning that until a user has performed a certain number of interactions, the system struggles to provide meaningful recommendations.

2.1.3.2 Content-based filtering recommender system

The word "content" in its name indicates that, unlike the previous method, it relies on descriptions of item features rather than collaborative data from users to provide personalized recommendations. For example, if a user has positively rated the film "Rocky Balboa", the content-based filtering (CBF) RS will be able to suggest other items with similar characteristics, such as the film "Creed". This type of RS can be an alternative in cases where information on other users' ratings is not available, making CF RS unusable.

The content-based method therefore has two main steps in its algorithm : first it creates a profile for each item by capturing the most important content features, and then it combines this with corresponding ratings or buying behavior to create personalized models for each user. These models predict whether the user will like items with unknown ratings.

This approach has a number of advantages : it can provide recommendations for new items even if we only have a limited amount of rating data on them. This is because the user may have rated other similar items, so the system can evaluate these items as interesting for the target.

However, there are also disadvantages. Firstly, there is a lack of diversity : since recommendations are based on specific item attributes, they can sometimes be obvious and lack originality. Additionally, items with less common attributes have a lower chance of appearing, which can be detrimental to the user experience and raise questions of fairness between items. Secondly, there is the ”cold-start” problem, as this RS requires a minimum amount of user evaluations to make accurate recommendations. A solution to this issue can be found by moving away from the traditional approach and allowing users to provide a few keywords or preferences in their profiles.

2.1.3.3 Demographic recommender system

As the name suggests, this type of RS uses users’ demographic data to generate recommendations. This data may include age, location, gender, marital status, or other socioeconomic characteristics. The approach is based on the idea that individuals with similar demographic profiles are likely to have similar interests. One advantage of this method is its simplicity, as it requires only basic demographic profiles.

However, it also has disadvantages: demographic data alone are not sufficient to capture the complexity of user preferences, and individuals from the same group may not necessarily share the same tastes. Additionally, specific and less common items may not be recommended, as demographic data are fairly general. This approach is often combined with others to compensate for its lack of precision, known as a hybrid approach.

2.1.3.4 Knowledge-based recommender system

As the name suggests, these RSs exploit domain-specific knowledge to provide personalized recommendations to users. Unlike CF approaches, these systems do not rely on historical user interactions, but use explicit rules, similarity functions, or specific cases to assess how certain item features align with user needs and preferences, thus determining the item’s usefulness to the user.

An example application could be a travel planning website : it would ask users to specify their preferred destination, budget, and travel dates. The knowledge-based RS uses explicit rules and similarity metrics to match users’ preferences with suitable travel packages and suggests personalized itineraries.

This RS has the advantage of excelling in the early phases of deployment, as it provides personalized recommendations based on user-specified requirements. However, if they are not equipped with learning components, they can be outperformed by other methods that exploit user interaction logs, as in the case of Collaborative Filtering RS.

Knowledge-based recommenders can be further classified based on the type of interface used to achieve their objectives :

(a) **Constraint-based recommender systems :** In this case, the user will give constraints on the attributes of the items they are interested in (e.g., lower or upper limits). Domain-specific rules are used to match these constraints to item attributes. Users can modify the constraints interactively according to the number and type of results obtained.

(b) **Case-based recommender systems :** Here, users identify specific cases as reference points, and the system employs similarity metrics based on item attributes to find items similar to those cases. Users can then interactively adjust the attributes to fine-tune the recommendation process.

Knowledge-based systems and content-based systems exhibit similarities since they heavily rely on item attributes for making recommendations. Nonetheless, their primary distinction lies in the input data they utilize : knowledge-based systems depend on user-specified requirements, whereas content-based systems learn from historical user behavior. This distinction is why they are considered separate approaches in the literature.

2.1.3.5 Hybrid recommender system

A hybrid RS combines two or more recommendation techniques to leverage the strengths of each. Generally, the CF technique is integrated with another method to address challenges posed by the ramp-up issue, such as cold-start or data sparsity. In his paper titled ”Hybrid Recommender Systems: Survey and Experiments”[7], Burke categorizes hybrid RS into seven distinct approaches :

(1) **Weighted :**

Among the various approaches to hybrid RS, weighted hybrids are the most frequently used. These systems calculate item scores by combining the output scores of various recommendation techniques through weighted linear functions. One of the first algorithms to use this approach was the P-Tango system, which combined CF and CBF techniques initially with equal weights, then gradually adapted the weights based on the confirmation or contradiction of user rating predictions.

The advantage of this approach is that it takes advantage of all the strengths of the system's recommendation techniques, making the recommendation process simple. It allows for performance evaluation and easy adjustment of weights if necessary. However, this approach makes the assumption that the different techniques used have a relatively similar value for all possible elements, which is not always the case. For example, as mentioned above, a CF suffers from the cold-start problem and would therefore provide weaker recommendations than a CBF for items with few user ratings.

(2) **Switching :**

Hybrid switching RSs are designed to dynamically switch between several recommendation techniques in order to take the most effective one depending on the context. For example, if a CF RS struggles to provide accurate suggestions for newly introduced items with limited data, this approach could perhaps switch to a content-based approach to better handle this case, thus drawing on the strengths and parrying the weaknesses of a technical RS. The only concern with this approach is that it adds more complexity as it requires switching criteria to be defined, which adds an extra layer of parameterization.

(3) **Mixed :**

Mixed-hybrid RS offer a pragmatic approach to hybridization, particularly when a wide range of recommenders can be integrated simultaneously. The results of several recommendation techniques are combined to create a consolidated list of recommended items.

An example of this approach that is well known in the literature is the PTV system. It uses content-based and collaborative methods to compile a customized TV program schedule. The system combines the recommendations of these techniques to propose a complete TV program.

(4) **Feature combination :**

Hybrid feature combination RSs takes the advantages of different recommendation techniques by integrating the outcomes of one RS as additional feature data for another. This approach can, for example, enhance the performance of content-based recommenders, which heavily rely a lot on item features, by incorporating collaborative data into their decision-making process.

There are two major advantages to this approach. Firstly, it mitigates the system's sensitivity to data sparsity issues that can arise in CF. By introducing collaborative data as features, it increases the information available to create recommendations. Secondly, it provides the system with insights into intrinsic similarities between features, which might not be obvious in a collaborative system.

(5) **Cascade :**

In contrast to previous hybrid approaches, the hybrid cascade method employs a multi-step approach. In this strategy, a first recommendation technique is used to generate an initial general ranking of potential candidates. Subsequently, a second technique refines the recommendations based on the previously created group of candidates.

This technique aims to mitigate the limitations of each recommender by leveraging their complementary strengths. The initial recommender, often a general technique such as CBF, generates a list of preliminary recommendations. These recommendations are then given as input to a subsequent recommender for refinement, usually CF. This cascade ensures that the final recommender, collaborative in our example, focuses on a smaller, more relevant group of items, thus improving the recommendations by managing the ”noise”.

(6) **Feature augmentation :**

This approach improves the quality of recommendations by incorporating the results of one recommendation technique into another in the form of additional feature data. Unlike the cascade method, which relies on a sequential refinement process, feature augmentation directly enriches the second technique's capabilities by incorporating the results of the first technique. This integration allows the second technique to benefit from the insights and predictions generated by the initial technique, thereby enhancing its overall performance. Unlike the cascade method, feature augmentation does not rely on a specific order of techniques but instead focuses on leveraging the complementary strengths of each method to improve recommendation quality.

(7) **Meta-level :**

This approach is somewhat similar to the previous one, but the difference lies in the fact that here, in the case of a meta-level hybrid, the complete model is given as input to the second algorithm. In feature augmentation, it's a learned model that generates features to be given to the second algorithm.

The great advantage of this approach is that the model supplied to the second RS technique serves as a condensed depiction of a user's preferences. This approach is particularly effective in the context of content/collaborative hybrids. Indeed, the collaborative algorithm can navigate this rich information more efficiently than with raw rating data.

2.2 Explainability on artificial intelligence

2.2.1 Interpretability Taxonomy

The interpretability of machine learning models involves various methods that can be classified according to key criteria, providing insights into the mechanisms used to improve model transparency and delineate their nature and scope. Here we describe the main categories in the field of machine learning interpretability :

2.2.1.1 Intrinsic vs Post hoc Interpretability :

This criterion distinguishes between the interpretability achieved through the inherent simplicity of the model (intrinsic) and the methods applied after the model has been trained (post hoc).

- **Intrinsic Interpretability** refers to the simplicity and transparency inherent in certain ML models, such as short decision trees or sparse linear models. As the name suggests, interpretability is built into the model itself, making it naturally understandable to humans.

- **Post-hoc Interpretability** involves applying methods to explain the decisions of a ML model after it has been built. These methods provide a better understanding of the model's behavior without being part of the model itself.

2.2.1.2 Model-specific vs Model-agnostic :

- **Model-specific interpretability** refers to techniques that treat the model as a grey-box, using features specific to a particular type of ML model. This allows us to understand how that specific model makes predictions.

- **Model-agnostic interpretability** implies methods that can be applied to any type of ML model. These methods consider the model as a black-box and operate independently of its architecture or complexity. This approach has the advantage of being universal and can explain the decisions of various types of models.

2.2.1.3 Local vs Global scope :

Another important criterion is the scope of interpretability methods. These methods can help explain individual predictions (local), the overall behavior of the model (global), or something in between.

- **Local Interpretability** focuses on understanding individual predictions or decisions made by the model. The goal is to explain why the model made a specific prediction for a single data point. This is particularly useful for end-users or stakeholders who are interested in understanding specific outcomes.

- **Global Interpretability** aims to provide a holistic understanding of the overall model behavior across the entire dataset. It involves elucidating how the model makes decisions generally, identifying the most influential features, and understanding the relationships between features and predictions. Global interpretability is crucial for model developers and regulatory bodies to ensure that the model operates fairly and ethically across all scenarios. Techniques such as feature importance analysis, partial dependence plots, and global surrogate models are employed to achieve global interpretability.

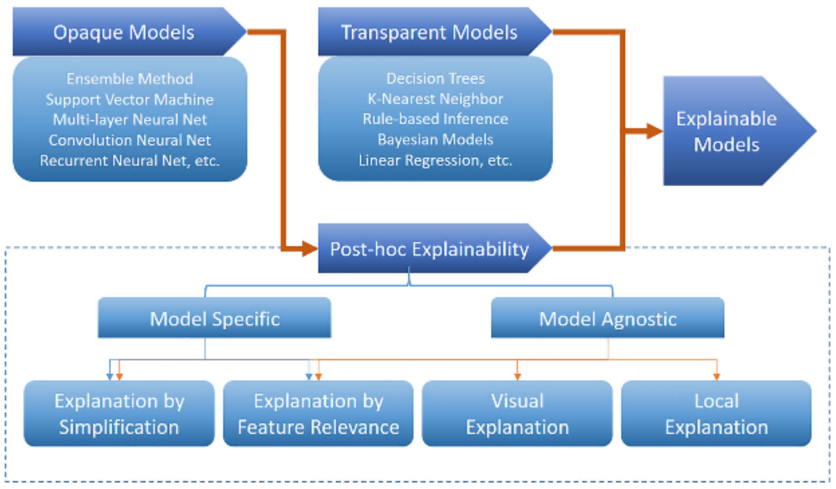


Figure 2.1: Overview of the explainable AI approaches. This figure comes from [3].

2.2.2 Transparent models

In the taxonomy of the previous section, these models belong to the intrinsic interpretability criterion and are designed to have an inherent simplicity that makes them understandable to human beings.

However, this simplicity generally comes at a cost and may sacrifice some level of predictive performance or modeling capacity. In the literature, we often speak of a trade-off between interpretability and model performance, which is circumstantial. For example, in scenarios involving huge datasets containing text or image data and where neural networks are suitable candidates for achieving the user's objectives, data scientists might shift their attention from intrinsic interpretability to post-hoc model explainability.

To get an overview of the different interpretable ML models that exist, note that a large number of new interpretable models are being created, making it challenging to cover them all comprehensively. However, we will explore the most well-known interpretable ML models in the literature, inspired notably by Christoph Molnar's book on interpretable machine learning [17] :

2.2.2.1 Linear regression

Linear regression is based on the assumption of a linear relationship between the input features and the target variable. The model can be represented by the equation :

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_p x_p + \epsilon$$

In this equation, y represents the predicted outcome, $x_1,\ x_2,\ ...,\ x_p$ are the input features, and ϵ denotes the error term capturing the discrepancy between the predicted and actual values. The aim of linear regression is to estimate the coefficients $\beta_0, \beta_1, ..., \beta_p$ for each input feature. These coefficients represents how each input feature contributes to the predicted outcome.

This model is easy for a user to understand by analyzing the coefficients associated with each variable : If the coefficient β_i is positive, this indicates that an increase in the corresponding input feature x_i implies an increase in the predicted result y.

2.2.2.2 Logistic regression

Logistic regression operates similarly to linear regression but is applied to classification problems, where linear regression does not excel. Instead of fitting a linear line or hyperplane, logistic regression uses the logistic function to transform the output of a linear equation into a range between 0 and 1. This logistic function is defined as :

$$\text{logistic}(\eta) = \frac{1}{1 + e^{-\eta}}$$

And here is what it looks like :

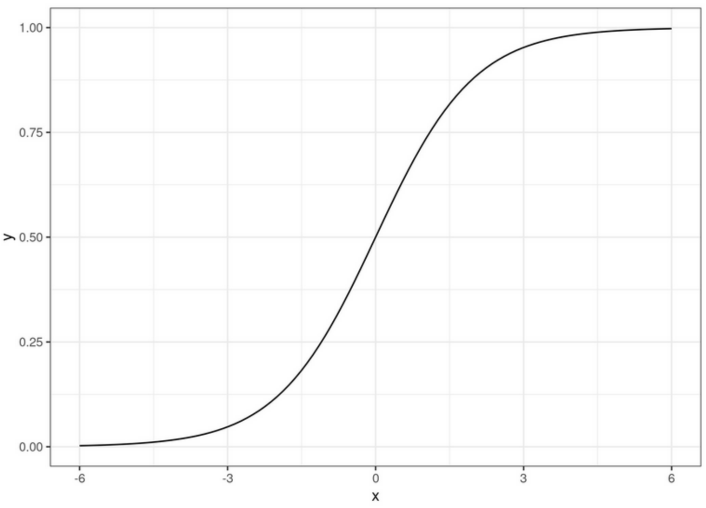


Figure 2.2: The logistic function : it outputs numbers between 0 and 1. At input 0, it outputs 0.5. This figure comes from [17].

Since logistic regression is used for classification, we look for probabilities between 0 and 1. Thus, we simply transform the linear equation from the previous section using the logistic function as follows :

$$P(y^{(i)}=1)=\frac{1}{1+e^{-(\beta_0+\beta_1x_1^{(i)}+\ldots+\beta_px_p^{(i)})}}$$

And so the interpretability is almost the same than as that for linear regression.

2.2.2.3 Decision trees

Unlike linear and logistic regression models, which struggle with nonlinearity and feature interactions, decision trees excel in such situations. These can be used for both classification and regression and work by repeatedly subdividing the dataset into multiple groups based on multiple features, forming a tree structure. Each internal node in the tree represents a decision made using a specific feature, and the terminal nodes indicate predicted outcomes.

The interpretation of such a process is easy to understand, as you can follow the path from root to leaf to understand the decision-making process.

2.2.2.4 RuleFit

RuleFit is a hybrid approach that combines the strengths of decision trees and linear models. It works as follows :

(1) First, it creates a set of decision trees that capture various rules segmenting the data based on feature values (each rule being a root-to-leaf path describing distinct conditions that influence predictions).

(2) Then comes the part specified to RuleFit, it transforms all these rules into a set of binary features representing the presence or absence of each specific rule.

(3) Finally, RuleFit uses a linear model on these binary features to learn how the presence of particular rules influences the final prediction.

The biggest advantage of this approach is that it is as easily interpretable as linear models, while allowing the integration of feature interactions as in decision trees.

2.2.2.5 Naive Bayes

Naive Bayes classifiers are based on applying Bayes' theorem with strong (naive) independence assumptions between the features. Mathematically, Naive Bayes estimates the probability of a class C_k given the observed features x as :

$$P(C_k|x) = P(C_k)\prod_{i=1}^{n} P(x_i|C_k)$$

In this equation, $P(C_k)$ represents the prior probability of class C_k, while $P(x_i|C_k)$ denotes the likelihood of feature x_i given class C_k. The model assumes that the features are conditionally independent given the class.

Despite its simplicity, Naive Bayes can perform surprisingly well and is easy to interpret, as it provides probabilities for predictions and allows the understanding of feature contributions.

2.2.2.6 k-nearest neighbors

The k-nearest neighbor (k-NN) algorithm is a straightforward method used for both classification and regression. For classification, k-NN assigns a class to a new data point based on the majority class among its k closest neighbors. For regression, it predicts the value of a new data point by averaging the values of its k nearest neighbors.

The performance of k-NN depends significantly on the choice of k and the method used to measure distance between data points.

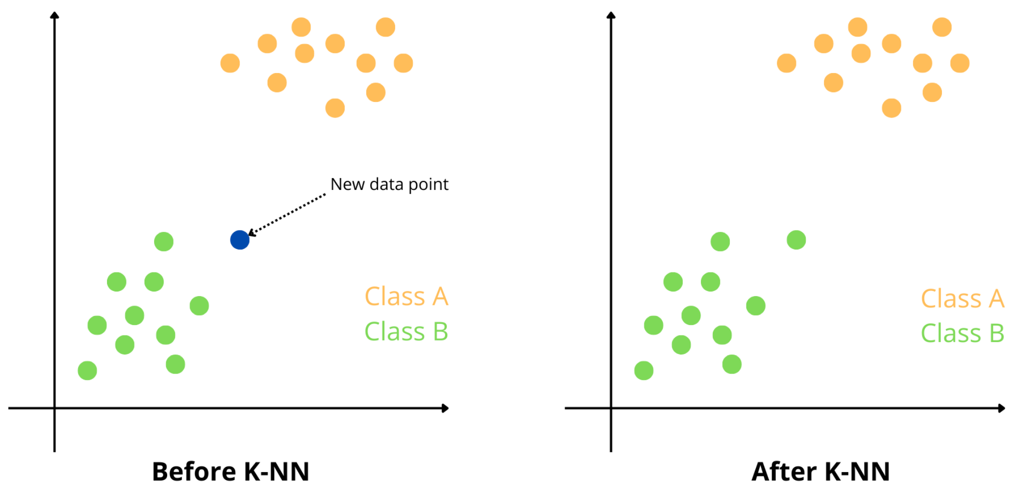


Figure 2.3: Illustration of K-nearest neighbor algorithm.

One of the unique aspects of k-NN is that it does not rely on a training phase to learn parameters. It memorizes the training data and makes predictions based on the stored instances. It means k-NN operates locally, focusing on the immediate neighborhood of each query point to make predictions. Consequently, k-NN offers excellent local interpretability, as the prediction for any given instance can be explained by examining the specific neighbors that influenced it.

However, k-NN lacks global interpretability because it does not produce a general model that summarizes the relationships in the data. Instead, each prediction is made independently, based on the local neighborhood. This makes k-NN highly interpretable in datasets with a small number of features, but it becomes complex and less transparent for datasets with many features.

2.2.3 State-of-the-art XAI methods

This section is based on the fairly recent paper (2021) entitled ”Explainable artificial intelligence: an analytical review” [3], which provides a state-of-the-art review of the most commonly used methods for enabling explainability.

2.2.3.1 Features-oriented methods

SHapley Additive exPlanation (SHAP) is a fairly inovative method of machine learning interpretability based on cooperative game theory that was introduced by Lundberg and Lee in 2017. Features are represented by "Shapley values", which reveal the impact of each feature on predictions. This approach offers both local and global interpretations, ensuring fairness and enabling meaningful comparisons between explanations. However, the transparency of this model may be affected.

Class Activation Maps (CAMs) are techniques designed exclusively for Convolutional Neural Networks (CNNs). They generate visual insights into the decision-making process of CNNs for image classification tasks. The CAMs process in an heatmap that overlays the original image highlighting the areas that significantly influence the network's predictions for different classes.

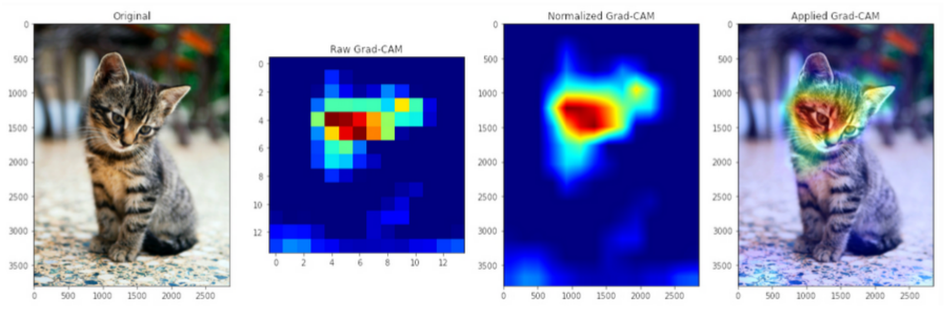


Figure 2.4: Example of usage of Class Activation Maps on CNN to classify cat pictures. This figure comes from a website [1].

An example of the application of CAMs to cat image classification can be found in the Figure 2.4. You can see which pixels in the image had the greatest influence on the classification, and thus get an explanation of the classification made. As CAMs are only specific to fully convolutional network architectures, we'll have to wait for the arrival of Grad-CAM or Grad-CAM++ to be able to apply this approach to all CNNs.

2.2.3.2 Global methods

Global methods in XAI aim to provide insights into the overall behavior of complex models, rather than focusing on individual predictions. These methods offer a holistic understanding of how a model makes decisions across various subpopulations or

[1]https://dlhr.de/8

datasets. One example of a global method is Global Attribution Mappings (GAMs), which utilize weighted conjoined rankings to explain a neural network's predictions across different subsets of data. By clustering similar local feature importances, GAMs summarize patterns detected in each cluster, providing a comprehensive explanation relevant to feature exploration among diverse subpopulations.

2.2.3.3 Concept methods

Concept Activation Vectors (CAVs), introduced by Kim et al. in 2021, are a designed approach to giving a global explanation of the inner workings of a neural network. To do this, human-interpretable features are mapped to high-level latent features that the neural network has learned. xplaining these associated concepts can reveal flaws in the model's decision-making process. These associations can reveal flaws in the model's decision-making process. In particular, this allows us to detect features that are considered significant when they shouldn't be.

2.2.3.4 Surrogate methods

Local Interpretable Model-Agnostic Explanations (LIME), introduced by Ribeiro et al. in 2016, is a robust approach for making local interpretability of complex black-box models. This method consists in perturbing the input features of a prediction and analyzing the resulting output variations. By examining these changes, LIME constructs a locally faithful and interpretable surrogate model that approximates the original model's behavior. While LIME significantly enhances interpretability, its effectiveness can be influenced by the selection of the neighborhood for perturbation and the choice of the surrogate model, impacting the quality of the explanations provided.

2.2.3.5 Local, pixel-based methods

Layer-wise Relevance Propagation (LRP) is an approach that was devised by Bach et al. in 2015 and aims to help explain the decisions of a complex multilayer neural network. It achieves this by tracing the output of the network back to the input data and uses rules to propagate relevance information across the layers. This enables it to create a heat map that shows the influence of each pixel in the model's prediction, making it easy for the user to interpret. n

2.2.3.6 Human-centric methods

The few methods presented above, while providing interesting explanations of machine learning predictions, fail to provide clear and understandable explanations for humans. They ultimately reveal only the edges of the ”black box” and offer limited post hoc insights into feature attribution. These explanations are still a long way from human reasoning, which attempts to evaluate similarities or to associate, to create analogies, etc. Angelov and Soares (2020) introduced a novel approach that views explainability as human-centric, recognizing that humans assess items holistically rather than just by individual features. An example of a human-centric approach is counterfactual explanation, which aims to provide insights and explanations for model predictions by generating hypothetical scenarios that show how small changes in input features would lead to different outcomes. A counterfactual explanation takes the form of the following example : "You were denied a loan because your annual income was £30,000. If your income had been £45,000, you would have been offered a loan."(Wachter, S., Mittelstadt, B., & Russell, C. (2018)[23]).

2.3 Explainability on recommenders

As discussed in the Section 2.1 dedicated to RSs, there is a wide variety of models for providing recommendations. With the continuous growth of research in this field, many hybrid approaches are emerging, making models increasingly complex and highlighting the need for explanations of their outputs. Explanation techniques used in recommender systems fall into two main categories: model-based and post-hoc.

Model-based explanations are integrated into the recommendation model itself, meaning that the model is designed to be interpretable from the outset. These explanations are generated as part of the model's decision-making process. This includes approaches like factorization-based, topic modeling, graph-based, deep learning, knowledge-based, and rule mining models. In contrast, post-hoc explanations are generated after the recommendations have been made, using external methods to interpret or justify the model's results. Post-hoc techniques do not modify the original model but provide insights into its decisions through additional analyses or auxiliary models. These are known as post-hoc or model-agnostic explanations.

A comprehensive survey on explainable recommendation models was conducted in the article by Zhang and Chen (2020) [25]. This survey offers a detailed overview of various approaches to explainable recommendations. Below, I will briefly summarize them :

2.3.1 Factorization Models for Explainable Recommendation

In explainable RSs, factorization models are among the most commonly used approaches. These models decompose user-item interaction matrices into latent factors for users and items, and predictions are made based on these factors. However, these factors are often not easily understandable to humans. This is why Explicit Factor Models (EFMs) were developed to enhance interpretability by associating these factors with more comprehensible characteristics, such as item attributes or user feedback. This association enables a more understandable and direct explanation by highlighting the item characteristics that led to the recommendations.

2.3.2 Topic Modeling for Explainable Recommendation

Topic modeling methods identify hidden topics within user-item interactions. One of the most well-known methods is Latent Dirichlet Allocation (LDA).

In the context of recommendation, it works as follows: By observing the interactions between users and items, the model detects underlying themes (for example, in the context of a book recommender, it might detect genres like ″adventure,″ ″action,″ ″history,″ etc.). It then associates each user and item on the platform with these themes (for instance, some users might be more interested in history, while some items might be categorized as cookbooks). Based on this, when the recommendation system is queried about the recommendations made, it can respond, for example: ″This book is recommended to you because it contains characteristics of a history book, and you seem to appreciate that″.

2.3.3 Graph-based Models for Explainable Recommendation

Graph-based models use graph structures to illustrate the relationships between users, items, and their attributes. By utilizing knowledge graphs or user-item interaction graphs, these models can trace and clarify recommendations by following the paths and links present in the graph, thus providing detailed explanations for the recommendations. For example, if a user has interacted with several books of a certain genre, the model can find other books of the same genre and recommend them. If an explanation is then requested, it can respond: ”This book is recommended to you because it is related to the history books you have appreciated.”

2.3.4 Deep Learning for Explainable Recommendation

Deep learning approaches utilize neural networks to model complex interactions between users and items. Techniques such as attention mechanisms and visualization of neural activations are employed to give explanations on these recommendations. These methods highlight important features or interaction patterns, which make the deep learning process more transparent and understandable.

2.3.5 Knowledge Graph-based Explainable Recommendation

Knowledge graph-based models incorporate structured information about items and their attributes into the recommendation process. This is done using a knowledge graph, through which the model identifies meaningful paths between user preferences and certain item characteristics. Then, to justify and explain its recommendations, the recommendation system can emphasize the major characteristics that guided it.

2.3.6 Rule Mining for Explainable Recommendation

Rule mining approaches work by extracting association rules from interactions between users and item attributes. These rules take the form ”If X, then Y”, where X represents one or more conditions reflecting user preferences, and Y represents the recommended items. They form the basis of explanations by providing simple and understandable reasons for recommendations.

2.3.7 Model-Agnostic and Post-Hoc Explainable Recommendation

The Model-agnostic and post-hoc approach represents the second category, where the RS is considered as a black box, and the explanation step of a recommendation does not rely on any internal feature of the model to make its decision. This approach utilizes techniques such as surrogate models or feature importance analysis. It provides additional insights to justify its decisions, as demonstrated by the counterfactual explanations approach presented in my thesis.

Chapter 3

Related work

The purpose of this section is to introduce the main paper upon which this thesis is based, titled ”Model-agnostic counterfactual explanations of recommendations” [14], co-authored by my thesis advisor, Mr. Sacharidis, and two other experts.

In their article, the authors propose a model-agnostic approach to provide explanations to users regarding why a RS suggested certain items. This approach is intended for both end-users of the platform and more experienced users, such as developers. The method relies on counterfactual explanations, which involve making slight changes to the input data to observe how these changes affect the model's predictions. Specifically, in the context of their study, the input data comprises the user's interaction history (such as clicks, ratings, and views), and the approach involves modifying this history by removing certain interactions.

Unlike other post-hoc approaches that train a surrogate model and require interaction data from all users, this method is designed to be privacy-preserving, as it only needs the history of the user seeking the explanation. This ensures the explanations are precise and directly relevant to the user's specific context.

However, since the approach is model-agnostic and does not have access to the internal workings of the model, the problem becomes a search problem within the user's history to find the shortest possible list of interactions to remove. The paper proposes several strategies to address this and compares them to more traditional methods such as exhaustive and random search. Additionally, the article considers the constraints of time and limited model queries by introducing the concept of a budget in the studied strategies. The budget represents the number of times the search strategy can query the model to test modified inputs and verify if the recommendation system's output meets the desired criteria.

3.1 Formal Problem Definition

Consider users who have *interacted* with a set of *items* I, where $n = |I|$. Given this input set I, a RS generates a ranked list of *recommendations* R (see Figure 3.1), comprising a selection of $m = |R|$ top-ranked items (e.g., if $m = 20$, it means the selection includes the top-20 ranked items). The central question users might ask is "Why was this particular *target item* $t \in R$ recommended to me?"

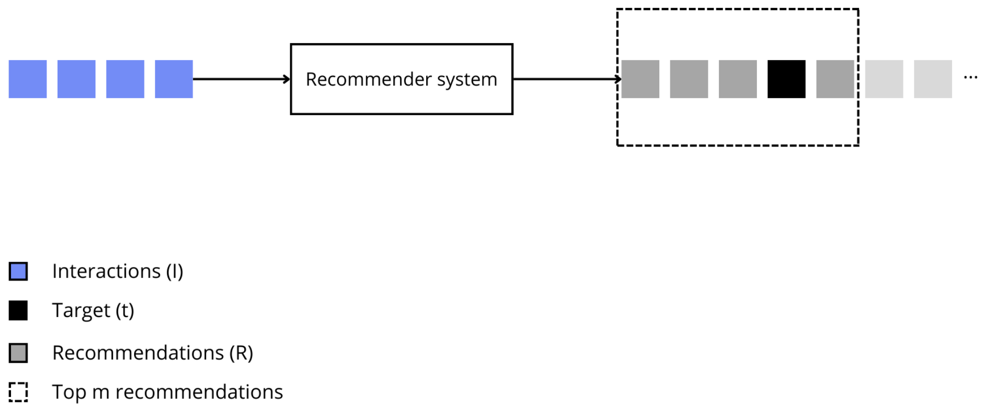


Figure 3.1: Illustration of the recommendation system

First, we denote the position of the target item in the ranked recommendations list as the *target position*. To address the users' queries regarding the *target item* t, the explanation mechanism will look for a subset of the interacted items, $E \subseteq I$. This subset constitutes the *counterfactual explanation* E and, if removed from the users' interactions ($I \setminus E$), would result in the RS generating a new list R' that no longer contains the target item, i.e., $t \notin R'$.

Furthermore, there may not be a single possible counterfactual explanation; several sets of elements may achieve the desired outcome. Section **??** will present quality measures for selecting the best option. In the meantime, we designate a set of elements that have yet to be evaluated for their potential to achieve the desired result of $t \notin R'$ as a *candidate counterfactual explanation*, denoted by $C \subseteq I$.

3.2 Approach

In addressing the problem at hand, the primary objective is to explore subsets $C \subseteq I$ (candidate solutions) comprising interacted items, with the aim of pinpointing a subset E. However, as the size of the users' interaction history n grows, traversing the entire search space of counterfactual explanation candidates becomes impractical. Due to time constraints and/or a limited number of calls to the RS, the strategies proposed in the article employ heuristics. This is achieved by introducing a restriction with a budget B that the strategies cannot exceed. This budget represents the number of times the search algorithm can evaluate a candidate C.

It is important to note, as highlighted in the paper, that human beings are more receptive to short explanations, making the size of the explanation (i.e., the size of the subset E) crucial to the quality of the explanation provided. The shorter and more concise the counterfactual explanation, the more feasible and understandable it will appear to the user.

To achieve these objectives, in addition to the budget, the authors have introduced two quality measures to calculate a score for each candidate explored and to guide the search within the strategy search space. The first measure is the normalized length:

$$l(C) = \frac{|C|}{|I|}$$

This measure represents the proportion of items in the candidate set compared to the total number of interacted items, ranging between 0 and 1 (excluding 0).

The second measure is called the impotence of a candidate and is defined as follows:

$$i(C) = \max\left(\frac{m - \text{rank}(t; C) + 1}{m}, 0\right)$$

In this equation, $\text{rank}(t; C)$ denotes the position of the target item t in the ranked recommendation list generated by the system when considering the set C as the new interactions of the users (i.e., new input to the model). This measure calculates the inefficiency of a candidate set C in removing the target item from the top m recommendation list. The closer the impotence value is to 1, the closer the target item is to the top rank (= first recommendation in the list). Conversely, if the target item is outside the top m-ranked recommendations, the impotence will take the smallest possible value of 0.

Therefore, it is desirable for a candidate evaluated by a strategy to have a low loss score, meaning both a low normalized length and low impotence. To find the best possible explanation in the candidate search space, the heuristic algorithm will have to strike a balance between the aspiration to seek a more concise counterfactual explanation E and the restiction to limit itself to the prescribed budget B.

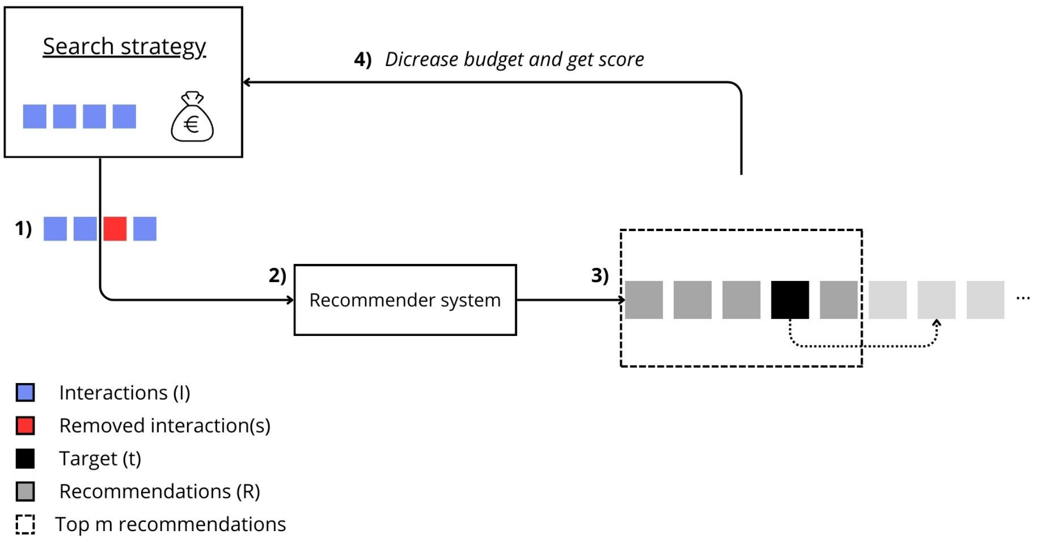


Figure 3.2: Illustration of an iteration of a strategy to find explanations to "Why questions".

The general idea of the algorithm is to operate over several iterations, with the number of iterations limited by the budget. As illustrated in Figure 3.2, each iteration consists of four parts:

1. **Selection of Candidate Set C :** The algorithm selects a candidate set C to be removed from the users' list of interactions. This selection follows one of the strate-

gies explained in the subsequent sections (Exhaustive Search, Random Search, Breadth First Search, Priority Search or Hybrid Search).

2. **Modification of the Input of the RS :** The algorithm provides this modified interaction set to the recommender system.

3. **Generation of Recommendations :** The RS produces a new list of recommendations R'.

4. **Evaluation and Budget Update :** We then analyze whether the target item has moved and whether the result meets the expected outcome, i.e., the target item is no longer in the top m recommendations. The new recommendations are evaluated using two quality measures: normalized length and impotence. Additionally, the remaining budget is decreased by 1.

This loop continues until the budget is exhausted, or for some strategies (like exhaustive search), it may stop earlier if an explanation is already found. The solution with the best score (i.e., the one with the lowest loss) is then returned.

One important observation we can make about the search space is that it can be ordered by inclusion : we start with the empty set, then consider sets containing a single interaction from the history, followed by all the distinct sets containing two interactions, and so on, until we reach the complete set I. To explore all these possible combinations, we can use graph traversal algorithms to identify a counterfactual explanation.

3.3 Strategies

This section aims to present the different strategies for exploring the search space to find a candidate. It begins by presenting two baseline strategies (Random search and exhaustive search) with which the new strategies proposed in the article will be compared and studied.

3.3.1 Random Search

The Random Search (Rnd) strategy randomly selects candidates in terms of both their cardinality and the chosen items. It continues until the budget B is exhausted, and then retains the counterfactual explanation E with the smallest size (explanation length), if such an explanation exists.

3.3.2 Exhaustive Search

This strategy (Exh) examines candidates by gradually increasing their cardinality: it first looks at all sets of size 1, then size 2, and so on. The search ends as soon as it finds a counterfactual explanation, since this will be optimal in terms of length, or when the budget is fully used.

3.3.3 Breadth First Search

The Breadth First Search (BFS) strategy operates in two phases. In the first phase, BFS incrementally constructs a candidate explanation by adding one item at a time to an initially empty set. For each expanded candidate, BFS queries the recommender system and explores subsets of decreasing cardinality within the current lattice level. The process stops when the budget is exhausted or the search space is fully traversed. BFS outputs the shortest counterfactual explanation based on target item rank, continuing to refine it in the second phase.

In the second phase, BFS investigates whether a subset of the initially identified counterfactual explanation can yield a shorter explanation. This entails a breadth-first search within the sub-lattice rooted at the counterfactual explanation. BFS prioritizes shorter and impactful counterfactuals, optimizing the search process while adhering to budget limitations.

3.3.4 Priority Search

The Priority Search (Pri) strategy aims to explore promising parts of the lattice. Candidates are assigned priority scores, a combination of their normalized length and impotence. The score is computed using a convex combination formula:

$$s(C) = \alpha \cdot i(C) + (1 - \alpha) \cdot l(C),$$

where $\alpha \in [0, 1]$ is a weighting factor.

During the search, candidates are added to a min-heap based on their priority score. In each iteration, Pri selects the candidate with the lowest score from the heap, exploring its neighborhood to identify more promising candidates. Pri considers subsets of cardinality $|C| - 1$ and supersets of cardinality $|C| + 1$ for each candidate in the neighborhood. For every explored candidate, Pri queries the recommender, calculates its priority score, and updates the heap. The strategy continues until the budget is exhausted, returning the shortest counterfactual explanation E if found.

3.3.5 Hybrid Search

The Hybrid Search (Hyb) strategy combines elements of exhaustive and priority search methods. It is designed to capitalize on the likelihood of finding short explanations in most cases. Hyb adopts an exhaustive search for candidates up to a predetermined small cardinality, such as 2. If a counterfactual explanation is discovered during this phase, the strategy terminates. Otherwise, each examined candidate's priority score is computed, and all candidates are added to the heap. From this point onward, Hyb follows the priority search approach, traversing the lattice until the budget is expended. In essence, Hyb extends the priority queue-based search by including all small-cardinality candidates. For our evaluation, we focus on candidates with up to two interacted items.

Chapter 4

Methodology

4.1 Introduction

As discussed in the Chapter 3, the paper [14] explores strategies to generate counterfactual explanations in response to ″Why″ questions, such as ″Why did I receive this item t in my recommendation list?″. The objective of my thesis is to extend this work by adapting the code to address questions known as ″Why-not questions″ in the literature [4]: **″<u>Why</u> did I <u>not</u> receive this item t in my recommendation list?″**.

To answer such questions, the explanation provided will remain counterfactual, showing what specific changes in the RS input data would have led to the desired result (i.e., t within the top-m of the recommendation list). However, the nature of the modifications to the input will differ from those presented in the Chapter 3: instead of removing items from the user′s interaction list, we will add items to it. The goal is to show users which item(s) they should have interacted with to have the desired item (t) in their recommendation list.

Although the nature of the modifications is different, the initial approach to the problem appears similar : we explore the search space using the previously implemented search strategies and modify the interaction list with the new subset chosen by the strategy.

However, a significant difference arises in the search space composition. This is now formed by all possible subsets of the items not interacted by the users, rather than subsets of interacted items. This difference generally expands the search space significantly, as the users' histories are typically much smaller than the list of existing products on a platform, resulting in a vast set of non-interacted items. The search space is therefore exponentially larger than the size of the list of non-interacted items.

On modern platforms like Amazon, where the number of available items runs into the millions, this can be problematic. To explore the candidates and find a suitable explanation in such a scenario, strategies will need more time/budget, which is impractical. A new approach is therefore needed to tackle this problem more effectively.

4.2 Approach

To explain the problem more formally, to answer "why-not questions", research strategies must select a candidate C (a subset) from the list of non-interacted items, denoted I^-, and add this candidate to I. The best candidate selected will then be the returned explanation E.

In the remainder of this thesis, we'll use the term "why scenario" to refer to the problem addressed by the reference article, and we'll use "why-not scenario" to represent the problem addressed in this thesis.

4.2.1 Sampling techniques

As explained earlier, the set I^- and the resulting search space is generally too large. The idea was therefore to reduce this set by retaining a certain number of items most likely to bring the target item t into R'. To achieve this, two sampling techniques were devised :

- The first technique involves using a similarity matrix between items, using the Jaccard distance as the metric. A similarity matrix is a square matrix where each element (i, j) represents a measure of similarity between elements i and j. Based on this matrix, we establish a ranked list of the items most similar to t. To do this, we retrieve the values in the positions (i, j_1), (i, j_2), ... (i, j_k) of the matrix, where i represents the index for the target item and $j_1, j_2, ...j_k$ represent the indexes of the items not interacted by the users. We then sort these values and select the desired number of items most similar to the target, increasing the chances that the RS recommends t since the added items resemble it.

- The second technique uses the RS model to retrieve items most similar to t. We give the target item t as an input to the RS, which returns a ranked list of the best items. We then take the number of top-items needed.

We denote as $I^{-'}$ as the new sample extracted from the set I^{-}.

Although these new sampling techniques seem to offer a way of solving the problem of the search space size, they also have some limitations :

1. The approach was designed to provide explanations in a model-agnostic way and to keep the data of each user private. Using a Jaccard similarity matrix between items, the approach loses these characteristics. To create such a matrix, it is necessary to use the interactions of other users to determine which items are likely to be similar to each other. Moreover, using the Jaccard similarity matrix only makes sense in the context of a collaborative filtering recommender system (see Section 2.1.3.1), making the approach no longer model-agnostic.

2. To overcome the first limitation, we might suggest using the second sampling technique that leverages the recommendation system, thus avoiding the problem. However, this second approach also has limitations: some types of RS, such as content-based ones (see Setion 2.1.3.2), suffer from the "cold-start" phenomenon. This technique is therefore not applicable to all RS and therefore not model-agnostic.

4.2.2 Adaptation of the use of the search strategies

Despite the search space for a candidate solution C being different from that in the "why scenario", it retains the same property of forming a partially ordered set (by inclusion). This means that the idea behind each strategy, presented in the chapter on related works, remains fundamentally the same.

From a coding perspective, this observation is interesting because it means that if we specify, using the function parameters, the set of items from which the search space will result (in this case I^-), we can reuse all the classes implementing the different strategies, adhering to the fundamental programming principle of DRY (Don't Repeat Yourself).

However, as mentioned in the Chapter 3, two measures are used to guide strategies in their search: impotence and normalized length. The normalized length remains the same (although some code adjustments are necessary). The size of the new candidate, even if it is added, must be as concise as possible :

$$l(C) = \frac{|C|}{|I|}$$

But impotence must be adapted. The original formula was:

$$i(C) = \max\left(\frac{m - \text{rank}(t; C) + 1}{m}, 0\right)$$

and now becomes:

$$i(C) = \max\left(\frac{\text{rank}(t; C) - m + 1}{m}, 0\right)$$

As a reminder, rank($t; C$) denotes the position of the target item t in the ranked recommendation list generated by the system when considering the set C as input.

This measure now assigns more loss when a possible solution does not move the target item within the top m recommendations. The further you are from the top m, the more malus it will receive.

This change also had to be made in several other places in the code because, for example, for the BFS strategy, it changes behavior when a solution has been found and seeks to explore in the opposite direction. However, now to be a solution, we must allow t to be in the top m.

4.2.3 General algorithm

To summarize, the algorithm will first retrieve the subset $I^{-'}$ and then follow a process very similar to that explained in Section 3.2: through several iterations, the number of which is limited by the budget B, it will choose a new candidate (following the strategy used) and evaluate it. If the new candidate has the best evaluation so far, then we keep it.

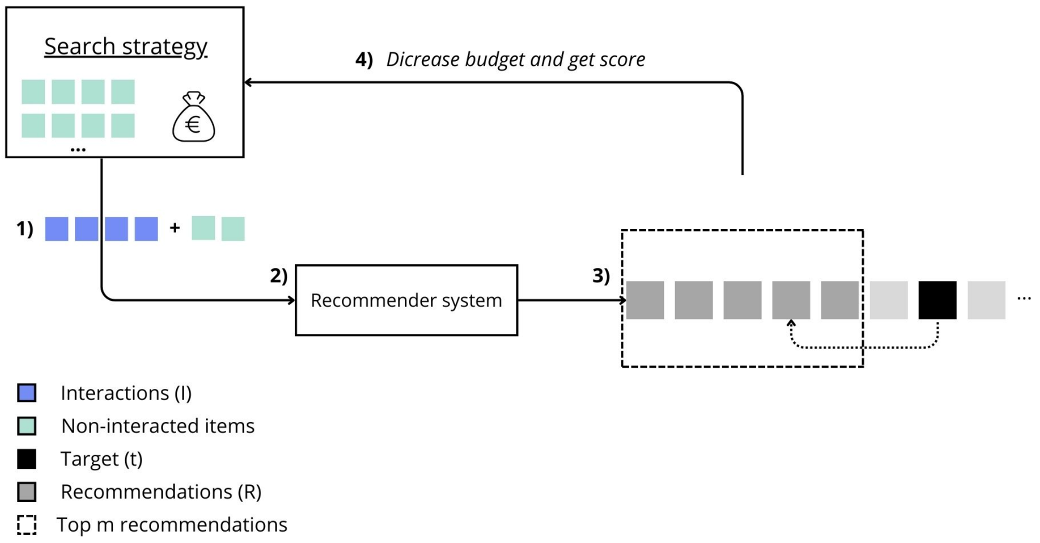


Figure 4.1: Illustration of an iteration of a strategy to find explanations to "Why not questions".

As illustrated in Figure 4.1, an iteration still consists of four steps, but some of them have changed slightly:

1. **Selection of Candidate Set C :** The search algorithm returns the next candidate selected from the subset of non-interacted items $I^{-'}$ and adds it to the user's interactions I. This selection follows one of the strategies explained in the Chapter 3 (Exhaustive Search, Random Search, Breadth First Search, Priority Search, or Hybrid Search).

2. **Modification of the Input of the RS :** This part remains unchanged; the algorithm gives the augmented list of interactions to the RS.

3. **Generation of Recommendations :** The RS generates a new list of R' recommendations.

4. **Evaluation and Budget Update :** We analyze the recommendations and calculate the score represented by the loss calculated based on the updated quality measures: normalized length and impotence. If the loss is the smallest encountered, then this candidate is kept as the counterfactual explanation for integrating the target item into the top-m recommendations. Additionally, the remaining budget is decreased by 1.

We loop until the budget B is exhausted or (for some strategies, such as exhaustive search) a solution is found. The solution with the best score (i.e., the lowest loss) is then returned as the counterfactual explanation.

Chapter 5

Experimental evaluation

This section is dedicated to presenting the experiments conducted during this thesis and the results obtained. It will first provide details on the environment and technologies used to carry out the experiments. Then, the first experiment will analyze the performance of the various heuristic strategies presented in Chapter 3 by comparing them in the "why not scenario" in terms of the percentage of time an explanation has been found and the length of these explanations. An evaluation will then be conducted to ensure the reliability of the search strategies adapted to work in the "why not scenario". Finally, an evaluation will also be carried out to verify the proper functioning of the two proposed sampling techniques and their approach to reducing the search space.

5.1 Datasets and Setting

As the recommendation model, we used the pooling representation recommender implemented by the Spotlight library [1] along with the MovieLens 1M dataset [2]. This dataset contains approximately one million ratings for 3952 different movies by 6040 anonymous users through the MovieLens platform [3]. For these experiments, we chose $m = 10$, which means that users only received the top-10 recommendations. To provide these recommendations, we only considered users with at least 20 interactions, and for each of them, we only kept the first 20 interactions to ensure consistent results in our

[1]`https://maciejkula.github.io/spotlight/`
[2]`https://grouplens.org/datasets/movielens/1m/`
[3]`https://movielens.org/`

experiments.

The implementations were carried out and executed using Jupyter Notebook with Python 3.9. They are accessible in the following GitHub repository : `https://github.com/SnewZz/Thesis`.

5.2 Experiments and results

5.2.1 Approach

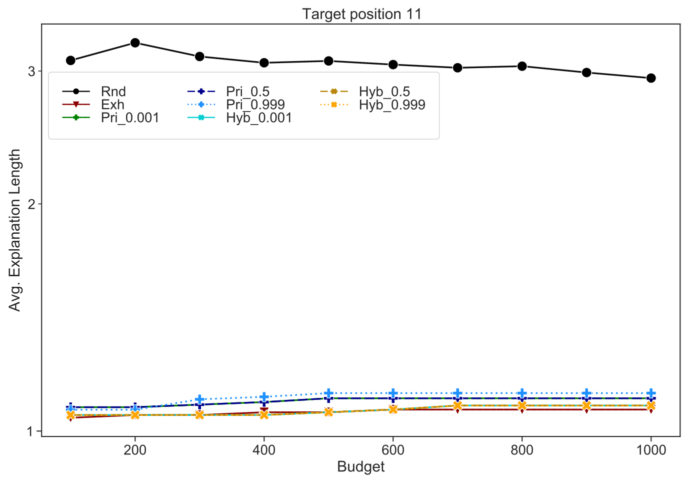
The first experiment conducted follows the approach of the main reference article [14]. This approach has been extended to the "why not scenario" and aims to compare and determine if heuristic strategies are more effective than a brute force algorithm or random exploration in finding counterfactual explanations in the search space. For this experiment, the priority search and hybrid search strategies were assigned several different alpha weights: 0.001, 0.5, 0.999, allowing for a comparison of the influence of the weight on the effectiveness of the strategies.

The scenario studied is the "why not scenario." As previously mentioned, this involves users who have received a list R of 10 recommendations and want an explanation for why a specific item t is not in their recommendation list. The designed algorithm will provide a counterfactual explanation, but to explore the search space (which in our experiment is of length 20 and chosen using the Jaccard similarity Matrix sampling technique), it is limited by a heuristic: the budget B. Our algorithm will therefore return the best explanation it has found until its budget is exhausted.

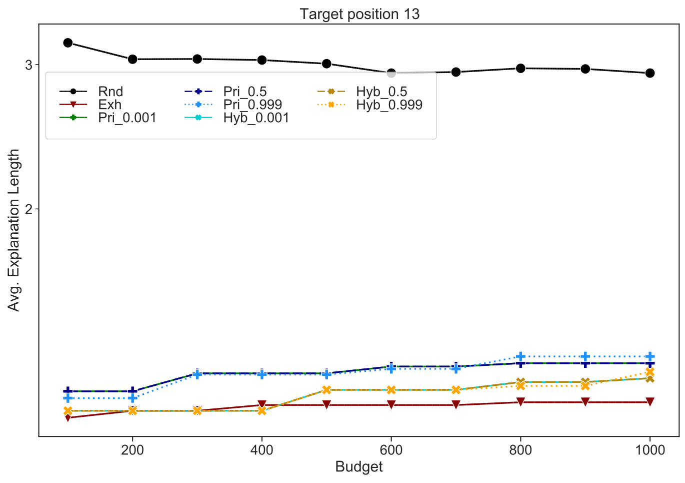
To study the performance of each strategy, two types of graphs are proposed : The first calculates the average size of the explanations found by each strategy. The second calculates the percentage of times an explanation was found.

In Figures 5.1a, 5.1b, 5.1c and 5.1d we run the algorithm for each of the proposed strategies and apply a series of predefined budgets to find a valid solution. For each of these budgets, we determine the average length of the counterfactual explanations, which are the sets of removed items that caused the target item to move beyond the

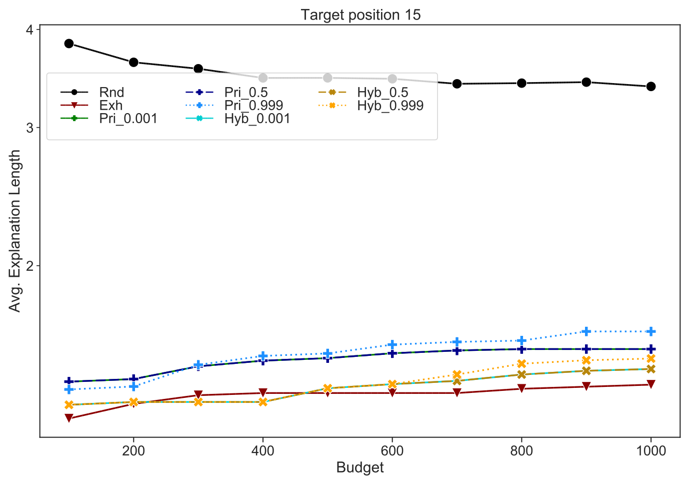
m-th position in the recommendation list within the given budget. Additionally, in Figures 5.2a, 5.2b, 5.2c, and 5.2d, we tally the number of recommendations for which an explanation is provided out of the total number of cases examined, again using the specified budgets.



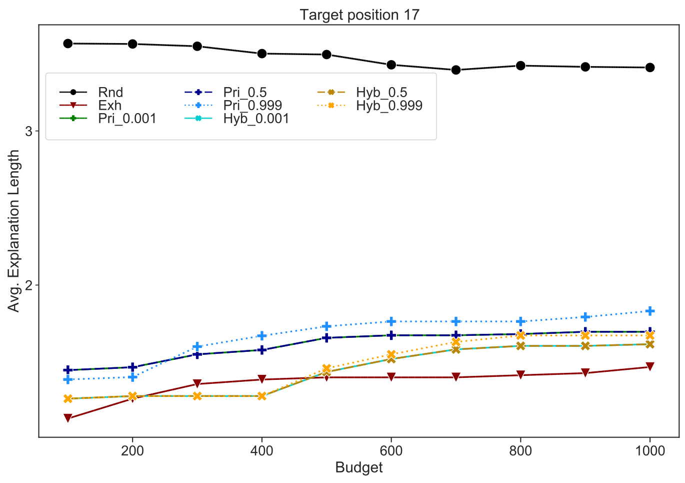
(a) Target item at position 11



(b) Target item at position 13

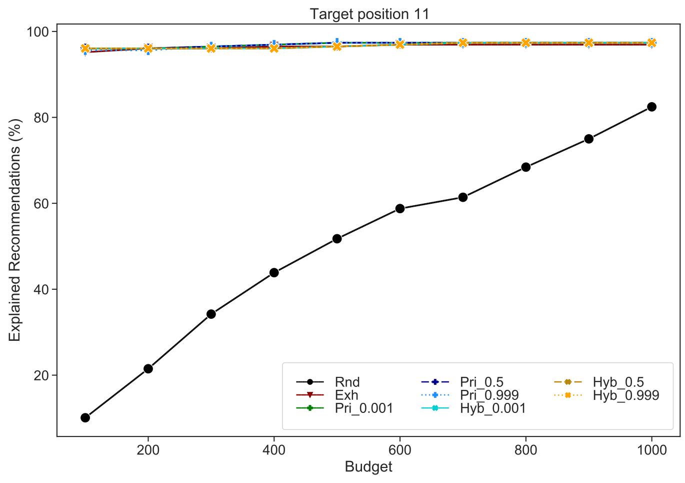


(c) Target item at position 15

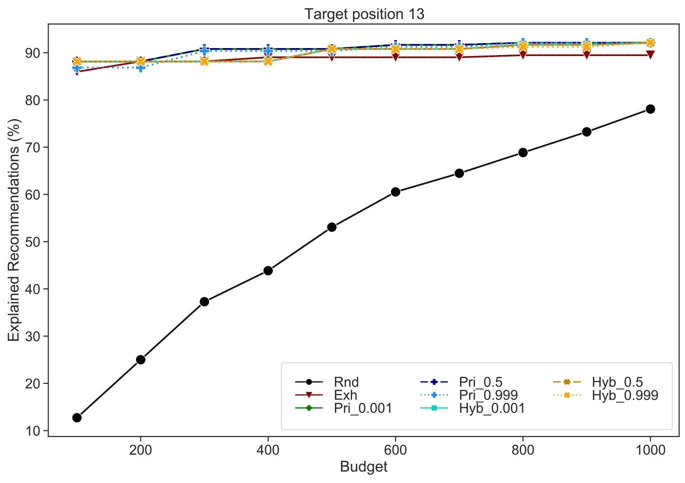


(d) Target item at position 17

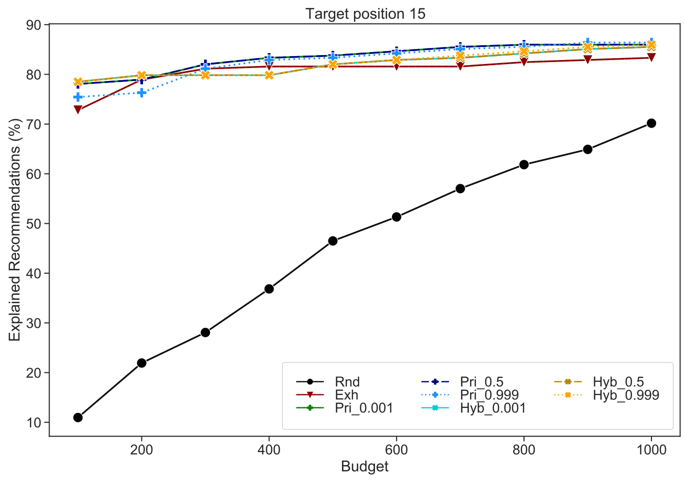
Figure 5.1: Plots of average explanation length vs. budget for different target positions



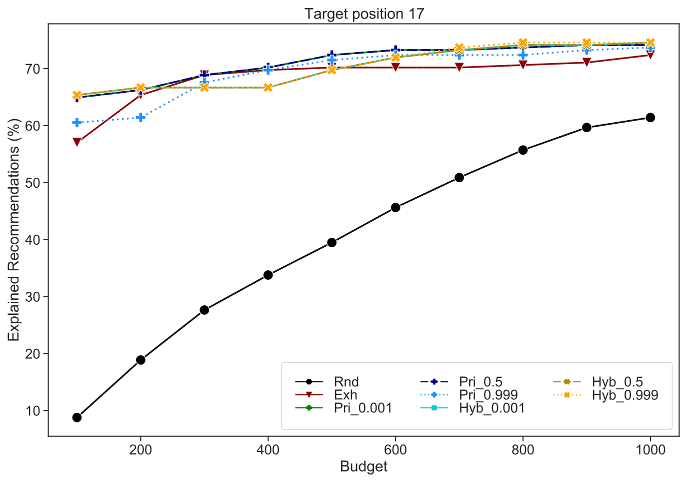
(a) Target item at position 11



(b) Target item at position 13



(c) Target item at position 15



(d) Target item at position 17

Figure 5.2: Plots of average explanation length vs. budget for different target positions

5.2.2 Observations

The first observation is that, as expected, there is a noticeable contrast between the random strategy and the other strategies. This can be explained by the less developed nature of the former compared to the others. As in the results of the reference article [14], this approach fails to deliver satisfactory recommendations, both in terms of the length of the explanations and the percentage of cases explained. From the graph, it is evident that this strategy requires a very large budget to provide concise explanations.

Next, a second observation that aligns with our intuition is that the further the target item's position is from the top-10, the more difficult it becomes for the strategies to find explanations. The explanation rate drops from nearly 100% for position 11 to about 65-70% for position 17. This necessitates a larger budget, and the length of the explanations increases, presenting more challenging cases where some search strategies struggle more, such as brute force search. However, as in the reference article's experiment, this strategy proves to be the most effective in terms of explanation length. Regardless of the budget, it remains relatively consistent (which is logical as it starts by exploring subsets of size 1, then size 2, and so on until it finds one). For the other strategies, we see that they are quite similar in terms of explanation length, but the priority search strategies with a higher weight tend to provide slightly longer explanations on average. In terms of the explanation rate, we notice that when the budget is low, priority search performs worse than the others, but with a higher budget, it aligns with the others and offers better performance than brute force.

Finally, if there is so little difference between the heuristic strategies and the contrast is already more noticeable in the results of the experiment conducted in the reference article, this can be explained by the quality of the search space in this scenario. Indeed, for this experiment, we used the Jaccard sampling method, which provides the 20 items most similar to the target item, making it much more difficult not to find a counterfactual explanation. This will be confirmed in the section dedicated to the evaluation of sampling techniques. In the article, the search space in the ”why scenario” can more quickly include items that have very little impact on the target item.

5.2.3 Limitations

It should be noted that due to a bug that I did not have time to fix, the BFS strategy could not be explored in the experiments. This bug prevented the correct execution of the BFS algorithm, and despite several troubleshooting attempts, the limited time allocated for this work did not allow for an adequate solution. Therefore, the results presented in this section do not include the performance of the BFS strategy. This limitation will need to be addressed in the future to obtain a complete evaluation of all strategies.

5.3 Test and evaluation of the code

To ensure the consistency of the results and verify that the code produces effective counterfactual explanations in the ”why-not scenario”, we conducted two different tests.

The first test examines the reliability and functionality of different strategies for selecting candidates. The second test evaluates the performance of the proposed sampling techniques and their ability to select a good search space.

5.3.1 Evaluation of the ”why not” scenario algorithm

The goal of this test is to evaluate the ability of strategies to bring the target item t back into the top-m recommendations. To do this, we need situations where t is outside the top-m but where a known combination of items, when added, would bring it back into the top-m (i.e., a counterfactual explanation). We could then drag these items into a list that would be completed by items considered to be as different as possible from t, identified using the Jaccard similarity matrix. This list forms the search space for testing strategies in the ”why-not scenario,” ensuring that a counterfactual explanation exists in the search space and must be found.

To generate such situations, the process was as follows : We take a user's interactions and feed them to the recommender system. We then choose a position in the top-m recommendations to designate the target item t. This allows us to determine the input that causes t to appear in the top-m. Using the "why scenario" algorithm (which is working well), we generate a counterfactual explanation, remove it from the input, and thus create a scenario where t is outside the top-m but we know the items needed to get t back into the top-m.

To ensure reliability, we needed a fairly large number of different situations. This test was conducted on the first 3036 user IDs, evaluating four different target positions (positions 1, 3, 5, and 7). After filtering for users with interaction lists of size 20, we were left with 1450 different users which gives 5800 different situations (1450 users * 4 target positions). To ensure that a counterfactual explanation is always found in the "why scenario," we used a brute force strategy with a budget of 1048576. Indeed, using the formula 2^n which allows us to know the number of subsets that can be found in a set of size n, we obtain 1048576 as the budget covering all possible subsets with the brute force strategy.

Due to the computational cost, the generation of these situations was recorded in a CSV file. This file includes 5800 lines representing the different scenarios. Each line records the user ID, target position, target item ID, original interaction list, and counterfactual explanation from the "why scenario."

Using this file, we can now evaluate each strategy in the "why-not scenario" for these 5800 different situations. The target position will be the one where the target item in the CSV is outside the top-m recommendations, and the search space will be the list of interactions that removed the target item from the top-m. We then analyze whether the strategy succeeded in finding an explanation and bringing the target item back into the top-m.

To evaluate each situation, this test will consider 4 different cases, in the first 3 it means that the new target position is in the top-m and are considered as a success and the last one means it is out of the top-m and then is a failure:

- Case 0: target item in the top-m and the new interactions are the same as the original interactions.

- Case 1: target item in the top-m and the new interactions are a subset of the original interactions.
- Case 2: target item in the top-m and the new interactions are different from the original interactions.
- Case 3: target item is not in the top-m.

The results obtained are as follows :

Table 5.1: Evaluation of the strategies in the ”why-not scenario”

	Case 0	Case 1	Case 2	Case 3
Exhaustive search strategy test	100%	0%	0%	0%
Random search strategy test	100%	0%	0%	0%
Hybrid search strategy test	100%	0%	0%	0%
Priority search strategy test	100%	0%	0%	0%

As we can see in the Table 5.1, the test seems to pass well, every strategy is able to find good counterfactual and choose the good candidates. As expected, the test show that all the situations fall in the Case 0 because as we used the worst jaccard item to complete the search space, the only items that could be selected are from the counterfactual explanations from the CSV.

5.3.2 Evaluation of the sampling techniques

The goal of this test is to evaluate the two proposed sampling techniques for reducing the search space, which is often too large in the ”why-not scenario.” To do this, we will reuse the CSV file generated in the previous test, starting from the situation where the target item is no longer in the top-m and we aim to bring it back. Thanks to the CSV file, we know a way to include t in the top-m, and thus the number of items to add to the interaction list to achieve this result (i.e., the length of the explanation). The idea now is to run the algorithm using the search space proposed by the sampling techniques and see if this approach can find explanations of equally good quality (which means of the same length).

For the search algorithm used, we chose the brute force method to compare the two sampling techniques directly, without being affected by changes due to using different search strategies. To assign a budget to this search strategy, we used the formula :

$$\sum_{k=0}^{x} C(n,k) = \sum_{k=0}^{x} \frac{n!}{k! \cdot (n-k)!}$$

This formula allows us to find the number of differents subsets of size smaller than k that we can find in a set of size n. In our experiment n equals 20 and k is the size of the solution contained in the CSV. This choice was made because the brute force strategy begins by exploring candidates of size 1, then size 2, and so on, stopping as soon as it finds an explanation or the budget is exhausted. By limiting the budget like this, it allows the strategy to explore only the necessary subsets of the search space. If, in any situation, it finds a solution larger than the one proposed in the CSV, we will simply consider this situation as a failure.

To evaluate each situation, this test will consider 4 different cases, the first 3 cases will be considered a success and the last one a failure:

- Case 0: the length of the explanation used to bring t into the top-m is smaller than that used to bring it out of the top-m.
- Case 1: the explanation used to bring t into the top-m is the same as that used to bring it out of the top-m.
- Case 2: the length of the explanation used to move t into the top-m is the same as that used to move it out of the top-m.
- Case 3: the length of the explanation used to move t into the top-m is greater than that used to move it out of the top-m.

Here are the results :

As we can see in the Table 5.2, the test works perfectly, showing that sampling techniques (both RS and Jaccard) are particularly effective: they enable strategies to find explanations. What's more, these explanations are even shorter than the one proposed

Table 5.2: Evaluation of the approach using sampling technique

	Case 0	Case 1	Case 2	Case 3
Jaccard sampling technique test	100%	0%	0%	0%
RS sampling technique test	100%	0%	0%	0%

in the CSV file, which is in fact an explanation derived from the search space formed by the user's original interactions. This leads to the following observation: The search space produced by sampling techniques is very good indeed.

Chapter 6

Future works

This section encapsulates potential enhancements to the current work presented in this thesis, along with novel ideas that emerged during its execution but could not be implemented due to time constraints. It serves as a repository of avenues for further exploration and development, offering insights into areas ripe for future research and innovation.

1. **Leveraging Differences and Similarities in Scenarios**

   Through this thesis, we noticed that the method to find a counterfactual explanation E in the scenarios of "why questions" and "why not questions" is essentially the same; the difference lies in what we do with E. In the first scenario, we seek to remove E from the input I given to the RS to exclude the target t from the top m recommendations. In the second, we add E to the input of the RS to include the target t within the top m recommendations.

   This difference arises because to adjust the position of the target item t, we modify the input of the RS by adding or removing items similar to the target. If the similarity of the input with t affects its position, then conversely, the dissimilarity can have the opposite effect. Thus, in the "why" scenario, we could remove items similar to t or add items dissimilar to t. Conversely, in the "why not" scenario, we could add similar items to the input or remove items too dissimilar to t.

   This observation is interesting because it could help in "difficult" cases, i.e., cases where the counterfactual explanation found is lengthy. For example, in the "why not" scenario, if adding similar items results in a lengthy explanation, removing dissimilar items might shorten it. This insight could lead to the development of new algorithmic approaches.

2. **Adaptive Sampling in the ″Why Not″ Scenario**

   Another idea that emerged during this thesis concerns the ″why not″ scenario. We introduced an additional sampling step to limit the search space by selecting only a defined number of the most promising items from the user′s non-interacted list (see Chapter 4). The exploration strategy then operates on this reduced search space to find a counterfactual explanation. An innovative idea would be to allow the algorithm to perform a kind of loop (see Figure) where, during its search, it can decide to restart the sampling method to generate a new search space. The decision to continue searching in the current space or to generate a new space could be made arbitrarily or by using algorithms such as the ant colony optimization algorithm.

3. **Applying Sampling Techniques to the ″Why″ Scenario**

   Finally, another improvement could be the application of the sampling technique used in the ″why not″ scenario to certain situations in the ″why″ scenario. If the user has a very large history and the search space size is dependent on it, integrating this sampling method to retain only the most relevant items (i.e., those most similar to the target t) could be beneficial.

Chapter 7

Conclusion

The primary objective of this thesis was to extend the work of my supervisor and two other researchers who presented a model-agnostic explanation mechanism for recommendation systems. This mechanism worked using counterfactual explanations and could only address "Why questions", such as "Why did I receive the target item t in my recommendation list?". The aim of this thesis was to adapt this mechanism to also answer "Why not questions", such as "Why did I not receive the target item t in my recommendation list?". This new scenario introduced a significant challenge: the search space became too large to find a good counterfactual explanation within a reasonable time frame. Therefore, in addition to adapting the strategies to function in this scenario, it was necessary to introduce sampling techniques to reduce the size of this search space.

To achieve this, two proposed techniques selected items similar to the target item so that if they were added to the user's interactions, the target item would have a higher chance of appearing in the recommendations. The first technique used a Jaccard similarity matrix to find similar items, while the second technique input the target item into the recommendation system to retrieve the most similar items from the perspective of the RS. Although these two approaches were very promising and provided a search space good enough to make some search strategies appear better, they also had some limitations: by using a Jaccard similarity matrix, the explanation mechanism lost its quality of being private and relying solely on the data of the user asking the question. As for the other technique, the recommendation system could suffer from a "cold start" problem, which hindered its proper functioning and caused our approach to lose its model-agnostic characteristic.

Throughout this thesis, we also conducted tests that ensured the proper functioning of the search strategies in this new scenario (except for the BFS strategy, which will require further investigation to identify the bug). In all situations generated by these tests, each strategy was able to find an explanation within the given budget. Finally, we replicated the experiment conducted by my supervisor in his article, which compared heuristic candidate search strategies (BFS, Priority and Hybrid) to two other baseline strategies (Random and Exhaustive). The observations were quite similar across the behavior obtained for several strategies.

In conclusion, several ideas for future work were raised:

- **Leveraging Differences and Similarities in Scenarios:** We observed that the method to find a counterfactual explanation E in both "why" and "why not" scenarios is essentially the same, differing only in the manipulation of E. This insight could lead to the development of new algorithmic approaches by using similarities and dissimilarities more effectively to generate shorter explanations in difficult cases.
- **Adaptive Sampling in the "Why Not" Scenario:** Introducing an adaptive sampling step to limit the search space showed promise. A future enhancement could involve allowing the algorithm to dynamically decide when to restart the sampling method to generate a new search space, potentially using strategies like ant colony optimization to make this decision.
- **Applying Sampling Techniques to the "Why" Scenario:** Applying the sampling technique used in the "why not" scenario to the "why" scenario could be beneficial, especially for users with extensive interaction histories. This would help in retaining only the most relevant items, thereby optimizing the search space.

This research has provided valuable insights into adapting model-agnostic explanation mechanisms to answer "Why not questions" in recommendation systems. We hope that the findings presented in this thesis will inspire further investigation and innovation in the field.

Bibliography

[1] C. C. Aggarwal et al. *Recommender systems*, volume 1. Springer, 2016.

[2] Amazon Web Services. Model explainability in aws ai/ml: Interpretability versus explainability, 2023.

[3] P. P. Angelov, E. A. Soares, R. Jiang, N. I. Arnold, and P. M. Atkinson. Explainable artificial intelligence: an analytical review. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 11(5):e1424, 2021.

[4] H.-M. Attolou, K. Tzompanaki, K. Stefanidis, and D. Kotzinos. Why-Not Explainable Graph Recommender. In *IEEE 40th International Conference on Data Engineering*, Utrecht (Netherlands), Netherlands, May 2024.

[5] E.-A. Baatarjav, S. Phithakkitnukoon, and R. Dantu. Group recommendation system for facebook. In *On the Move to Meaningful Internet Systems: OTM 2008 Workshops: OTM Confederated International Workshops and Posters, ADI, AWeSoMe, COMBEK, EI2N, IWSSA, MONET, OnToContent+ QSI, ORM, PerSys, RDDS, SEMELS, and SWWS 2008, Monterrey, Mexico, November 9-14, 2008. Proceedings*, pages 211–219. Springer, 2008.

[6] J. Beel, B. Gipp, S. Langer, and C. Breitinger. Paper recommender systems: a literature survey. *International Journal on Digital Libraries*, 17:305–338, 2016.

[7] R. Burke. Hybrid recommender systems: Survey and experiments. *User modeling and user-adapted interaction*, 12:331–370, 2002.

[8] E. Çano and M. Morisio. Hybrid recommender systems: A systematic literature review. *Intelligent data analysis*, 21(6):1487–1524, 2017.

[9] R. Confalonieri, L. Coba, B. Wagner, and T. R. Besold. A historical perspective of explainable artificial intelligence. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 11(1):e1391, 2021.

[10] F. Doshi-Velez and B. Kim. Towards a rigorous science of interpretable machine learning, 2017.

[11] C. A. Gomez-Uribe and N. Hunt. The netflix recommender system: Algorithms, business value, and innovation. *ACM Trans. Manage. Inf. Syst.*, 6(4), dec 2016.

[12] F. M. Harper and J. A. Konstan. The movielens datasets: History and context. *ACM Trans. Interact. Intell. Syst.*, 5(4), dec 2015.

[13] D. Jannach, M. Zanker, A. Felfernig, and G. Friedrich. An introduction to recommender systems. *New York: Cambridge*, 10:1941904, 2011.

[14] V. Kaffes, D. Sacharidis, and G. Giannopoulos. Model-agnostic counterfactual explanations of recommendations. In *Proceedings of the 29th ACM conference on user modeling, adaptation and personalization*, pages 280–285, 2021.

[15] G. Linden, B. Smith, and J. York. Amazon.com recommendations: Item-to-item collaborative filtering. *IEEE Internet computing*, 7(1):76–80, 2003.

[16] T. Miller. Explanation in artificial intelligence: Insights from the social sciences, 2018.

[17] C. Molnar. *Interpretable Machine Learning*. 2 edition, 2022.

[18] I. Nunes and D. Jannach. A systematic review and taxonomy of explanations in decision support and recommender systems. *User Modeling and User-Adapted Interaction*, 27:393–444, 2017.

[19] E. Pitoura, K. Stefanidis, and G. Koutrika. Fairness in rankings and recommendations: an overview. *The VLDB Journal*, pages 1–28, 2022.

[20] F. Ricci. Recommender systems: Models and techniques., 2014.

[21] F. Ricci, L. Rokach, and B. Shapira. *Recommender Systems: Techniques, Applications, and Challenges*, pages 1–35. Springer US, New York, NY, 2022.

[22] B. Smith and G. Linden. Two decades of recommender systems at amazon.com. *IEEE Internet Computing*, 2017.

[23] S. Wachter, B. Mittelstadt, and C. Russell. Counterfactual explanations without opening the black box: Automated decisions and the gdpr. *Harv. JL & Tech.*, 31:841, 2017.

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

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

Appendices

For additional resources and access to the code repository, please visit my GitHub page:

https://github.com/SnewZz/Thesis