**DOUGLAS LUAN DE SOUZA**

# ALGORITHMS FOR RESPONSIBLE EXPLANATION OF RECOMMENDATIONS

São Paulo
2024

**DOUGLAS LUAN DE SOUZA**

# ALGORITHMS FOR RESPONSIBLE EXPLANATION OF RECOMMENDATIONS

## Corrected Version

Dissertation presented to the University of São Paulo's Polytechnic School in order to obtain the title of Master of Science.
Program: Electrical Engineering.

Concentration Area:
Computer Engineering

Advisor:
Fabio Gagliardi Cozman

São Paulo
2024

Este exemplar foi revisado e corrigido em relação à versão original, sob responsabilidade única do autor e com a anuência de seu orientador.

São Paulo, 29 de Janeiro de 2024

Assinatura do autor: Douglas Luan de Souza

Assinatura do orientador:

Catalogação-na-publicação

de Souza, Douglas Luan
ALGORITHMS FOR RESPONSIBLE EXPLANATION OF RECOMMENDATIONS / D. L. de Souza -- versão corr. -- São Paulo, 2024.
136 p.

Dissertação (Mestrado) - Escola Politécnica da Universidade de São Paulo. Departamento de Engenharia de Computação e Sistemas Digitais.

1.Interpretabilidade e Explicabilidade (XAI) 2.Aprendizado de Máquina 3.Sistemas de Recomendação I.Universidade de São Paulo. Escola Politécnica. Departamento de Engenharia de Computação e Sistemas Digitais II.t.

# ACKNOWLEDGMENTS

I would like to acknowledge at first the invaluable help my family has given to me all this lifetime. First of all, my grandmother Maria Ferreira Mendes and my mother Maria de Lourdes Ferreira de Souza. Then, I would like to extend it to my brother Guilherme F. S. Sobrinho, my late grandfather José Martins de Souza, all my uncles, aunts and cousins. You've all witnessed a great deal of my struggle and victories and believed in me all along.

An immeasurable force that has been following me in each of my ambitions is the help of strangers, at first, who I today have the honor to call friends. It is certainly a difficult task to name them all and I would not dare to try to. But the help from my teachers Daladiê, Adriano, Raquel, Gleysson, from my former professor and supervisor Deborah Faragó, from my friends Cleber Santos, Matheus Portari, Diego da Cruz and their families. Also Regina Cerqueira, Gerald Thomson, Mathias Haake, Anna Haake, Christiana Müller and Matthias Knälmann who have helped my a great deal in my learning opportunities in Brazil, Australia and Germany. I could not be more grateful.

I would also like to express my gratitude towards my supervisior Fabio G. Cozman and my research colleague Gustavo P. Polleti for the respective supervision and collaboration. Likewise, I am grateful to the University of São Paulo (USP), the Federal University of the Jequitinhonha and Mucuri Valleys (UFVJM) and the University of Sydney for being part of my formal education. I extend my gratitude to the taxpayers who having been helping fund these institutions.

This work was carried out with the support of Itaú Unibanco S.A.; the author has been supported by the Itaú Scholarship Program (PBI), linked to the Data Science Center (C2D) of the Escola Politécnica da Universidade de São Paulo. The supervisor has been partially supported by the Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq), grant 312180/2018-7. This work has been partially supported by the Fundação de Amparo à Pesquisa do Estado de São Paulo (FAPESP), grant 2019/07665-4. We acknowledge support by CAPES finance code 001. We are also grateful to the Center for Inovation at Universidade de São Paulo (InovaUSP) for hosting our lab.

# ABSTRACT

Recommender Systems have been increasingly deployed as part of digital products such as e-commerce and social networks. Part of the reason for their success is their ability to predict which items the user will like. However, there are scenarios where their recommendation may not be in the best interest of the user. It could be because the recommendations reinforce a habit the users do not want, such as spending hours watching short videos, or because it suggests a product that does offer the best cost-benefit ratio. In these cases, it would be helpful for the users to understand the reasons behind the recommendation. Furthermore, it would be in the interest of the user to know what are the drawbacks of taking the recommendation. In this work, we propose methods for generating *reasons for* and *reasons against* a given recommendation, based on Snedegar theory of practical reasoning. We demonstrate that these methods are feasible in the context of education through a course recommender.

**Keywords** – Explainable Artificial Intelligence (XAI), Machine Learning, Recommender Systems.

# RESUMO

Sistemas de Recomendação tem sido cada vez mais presentes em produtos digitais como e-commerce e redes sociais. Um das razões para o seu sucesso é a sua crescente capacidade de predizer quais itens irão agradar seus usuários. No entanto, há cenários onde as recomendações podem não estar sendo geradas segundo os interesses dos usuários. Isto pode acontecer porque o sistema incentiva que o usuário passe horas assistindo vídeos curtos, ou porque ele recomenda algum produto que não tem a relação custo-benefício que seria melhor para o cliente. Nestes casos, seria desejável que os usuários pudessem entender as razões que levaram à recomendação. Além disso, usuários se beneficiariam ao compreender quais são as desvantagens de seguir a recomendação dada. Neste trabalho, propomos métodos para geração de *razões a favor* e *razões contra* uma determinada recomendação, baseados na teoria de Snedegar para raciocínio prático. Mostramos que estes métodos são aplicáveis na prática em um contexto educacional, usando um sistema de recomendação de disciplinas.

**Palavras-chave** – Interpretabilidade e Explicabilidade (XAI), Aprendizado de Máquina, Sistemas de Recomendação.

# LIST OF FIGURES

# LIST OF TABLES

# LIST OF ACRONYMS

| | |
|---|---|
| RS | Recommendation System |
| RecSys | Same as RS \| The ACM Conference on Recommender Systems |
| CRS | Conversational Recommender System |
| ML | Machine Learning |
| XAI | Explainable AI |
| KB | Knowledge Base |
| KG | Knowledge Graph |
| KE or KGE | Knowledge Graph Embedding |
| USP | Universidade de São Paulo |
| NLU | Natural Language Understanding |
| NLG | Natural Language Generation |
| DM | Dialogue Management |
| S1 | Reasons Against Algorithm - Scheme 1 |
| S2 | Reasons Against Algorithm - Scheme 2 |
| S3 | Reasons Against Algorithm - Scheme 3 |
| S4 | Reasons Against Algorithm - Scheme 4 |
| S5 | Reasons Against Algorithm - Scheme 5 |
| SLR | Stochastic Logistic Regression |
| CPU | Central Processing Unit |
| URL | Uniform Resource Locator |
| DFS | Depth-First Search |
| PRA | Path Ranking Algorithm |
| JSON | JavaScript Object Notation |
| HTTP | HyperText Transfer Protocol |
| MAUT | Multi-Attribute Utility Theory |
| LIME | Local Interpretable Model-agnostic Explanations |
| PRED | Explanation algorithm that uses the KG for predictions |
| GDPR | European General Data Protection Law |
| FAccTRec | Fairness, Accountability and Transparency in RecSys |

# LIST OF SYMBOLS

| | |
|---|---|
| $\mathcal{G}$ | Knowledge Graph |
| $i$ | Arbitrary recommended item |
| $\mathcal{I}$ | Set of $i$ |
| $e$ | Entity in a graph |
| $\mathcal{E}$ | Set of $e$ |
| $r$ | Relation in a graph \| rating for an item to be recommended |
| $\mathcal{R}$ | Set of $r$ |
| $\pi$ | Path representing an explanation in a graph |
| $\Pi$ | Set of $\pi$ |
| $u$ | User \| user information object |
| $\mathcal{U}$ | Set of $u$ |
| $r(u, i)$ | Affinity score between item $i$ and user $u$ |
| $h$ or $e_h$ | Head entity |
| $t$ or $e_t$ | Tail entity |
| $f_r$ | Plausibility score function |
| $\mathbb{T}$ | Vector, or list, containing plausibility scores |
| $g_r$ | Classifier that returns the tail entity with the highest plausibility score |
| $\Theta$ | Knowledge embedding parameters |
| $\hat{\mathbf{z}}_k$ | The kth perturbed sample of entities |
| $\mathcal{Z}$ | Set of $\hat{\mathbf{z}}_k$ |
| $R_h$ | Set of recommendations for entity $h$ |
| $\phi$ | Interpretable representation for an entity embedding \| set of *reasons for* |
| $\gamma$ | Explanation search function |
| $O$ | Big-O notation for algorithmic complexity |
| $\mu$ | Population mean |
| $\Gamma$ | Explanation format Gamma |
| $K$ | Explanation format Kappa |
| $\Xi$ | Explanation format Xi |
| $\overline{e}$ | Average effectiveness |
| $\mathsf{H}(X)$ | Entropy of $X$ |

# TABLE OF CONTENTS

# 1 INTRODUCTION

*"Any sufficiently advanced technology is indistinguishable from magic."*

-- Arthur C. Clarke

Contemporary society heavily depends on computers and digital information for various aspects of our daily lives, including industry, education, entertainment, and communication. The progress of hardware and software has been rapidly advancing since the completion of the first programmable general-purpose electronic digital computer, ENIAC (Electronic Numerical Integrator and Computer), in February 1946 [22, 8].

While these machines were at first aimed at research institutions and large companies, they went on to be part of virtually every household, as a desktop, a laptop or a smartphone. Furthermore, internet access has improved with ever higher coverage and bandwidth availability, leading to an expansion of the number of devices, specially smartphones [45]. Although the world wide web started as distinctively static by presenting the same content to all users, regardless of preferences, it evolved to be more and more personalized [18]. Today, user-fit content selection is a fundamental tool to deliver the best possible experience to users.

Contemporary practices prioritize the implementation of user-centric content selection as an essential tool in delivering tailored and customized experiences. One of the tools for delivering an enhanced customer experience using personalization is the Recommender System (RS), also referred to as Recommendation System. Such systems help humans in their decision-making process, i.e., to cope with the need of choosing from several possibilities of items where they do not have enough knowledge to decide or filter options. Although RS research emerged as an independent area in the mid-1990s, as the web was gaining popularity [6], they reached commercial applications by the end of the 1990s with the boom in electronic commerce (e-commerce). Since then, their technology has been widely utilized not only in e-commerce systems but also by media and content firms as part of the services they provide [65].

In the early 2000s, RSs already achieved substantial popularization, being the backbone of very successful businesses. For instance, Amazon.com, a global e-commerce platform, was one of the pioneers in this field, having launched its recommender system back in 1998, using an algorithm called collaborative filtering. Amazon has kept Recommender Systems as central tools in their e-commerce [69]. Another notable example is Facebook (originally thefacebook.com), a social network founded in 2004 that initially recommended friendships with people the user might know or be interested in connecting with. There is currently a multitude of companies relying on RSs as a central part of their business, such as NetFlix, Apple TV and Disney Plus for video streaming, YouTube, TikTok and Kwai for video recommendation, MercadoLivre, Americanas, Shopee and Shein for e-commerce, Tinder, Bumble, Umatch, ElitePartner for dating. Currently, besides the commercial success, RSs continue to be an attractive research area.

RSs can be roughly classified into *content-based*, *knowledge-based*, *community-based*, and those based on *collaborative filtering*. In a content-based system, items are recommended based on the similarities to those items previously chosen by the user. As for knowledge-based strategies, the RS recommends according to expert-given rules about user behavior. Community-based strategies employ information from social networks in which users engage to provide useful recommendations. In a collaborative filtering approach, the driver of item recommendation is the similarity of profiles among users (i.e. the systems recommend items that a similar user has chosen). In practice, RS developers frequently employ a combination of diverse strategies. For example, TikTok uses video title, audio, tags and user watch and upload history through a hybrid approach with collaborative filtering and content-based recommendation for personalizing the content the user will get recommended [82].

Among many different techniques for building RSs, one constant is that they rely on Artificial Intelligence (AI) tools to achieve their goals. The field of AI aims to build intelligent entities capable of perceiving, comprehending, and manipulating the world themselves. There are diverse approaches that have been tried in AI, each with a different degree of success [67]. In recent years, a particularly successful branch of AI has been Machine Learning (ML) [42, 12] which has been broadly applied, for instance to study human and animal behavior, drug discovery, surveillance, autonomous vehicles and protein folding [75, 74, 39, 35].

ML itself is classified into Supervised Learning, Unsupervised Learning and Reinforcement Learning. Supervised learning uses labeled data to train a model that should then find labels for newly input data. This division of ML is further subdivided into classification and regression, depending on the output being a discrete or continuous value, respectively. Unsupervised Learning, on the other hand, classifies data without labeling them. Reinforcement Learning is concerned with finding the optimal action to take in a game-like situation where the agent has aims to maximize the reward it receives [9]. Note that reinforcement is the term used in psychology for what is otherwise, e.g. in Economics, known as incentive.

ML techniques have allowed RSs to achieve ever higher prediction accuracy. This measure has been used to evaluate RS performance, with some prediction engine being at the base of the large majority of recommenders [65]. This metric may be measured by different mathematical expressions, but the concept it captures is simple: how well the recommender predicts that users will follow the recommendations they receive.

To achieve higher levels of accuracy, state-of-the-art RSs utilize various models in which non-observed variables are responsible for capturing patterns [28, 30, 33]. For example, matrix factorization techniques are applied to reduce the dimensionality of the data by identifying low-dimensional "latent factors" that may be combined [37]. The most popular latent model to this date is represented by embeddings, in which entities are mapped to numerical vector spaces where the similarity between entities is translated to distance between vectors [51].

The main inconvenience of latent models resides in the difficulty of interpreting them. Similar to other popular machine learning models, e.g. deep neural networks, embeddings are "black-boxes" that produce output by processing large quantities of data, without the possibility of easily mapping the relation between the input values and the resulting vectors. The very problem of low interpretability in highly accurate RSs reveal that this higher accuracy is not cost-free. As pointed out in [20], "The problem is that a single metric, such as classification accuracy, is an incomplete description of most real-world tasks".

Because RSs work in a way that requires them to communicate with humans to be helpful, they must be interpretable to be socially acceptable [48]. More importantly, humans need to understand the motivations behind other agents, such as RSs, so as to know whether their interests are aligned or conflicting [66]. But how to make a RS interpretable without losing its accuracy?

One solution to that problem is to generate explanations for the black-box so that humans can have a better understanding of it. Because some machine learning models are intrinsically interpretable, e.g. linear regression, we may use them to generate the explanations for "black-box" models. This leads to post-hoc explanations; that is, explanations are generated after the non-interpretable model is trained. Inputs and outputs are used to train an interpretable model that is then used to explain the original, more complex one [48].

Explanation generation for machine learning models is a topic of great interest in AI, both to allow users to detect possible flaws in those models and to increase the degree of user trust in the "black-box" systems [26, 25].

In this work, we study and develop explanations for a RS in the education domain using a post-hoc approach. Communication between students and the RS happens through a user-friendly chatbot interface. This interface allows students to input text queries and to receive textual responses displayed on the screen. Furthermore, the system offers an interactive feature that enables users to inquire about reasons for or against a recommended course.

## 1.1 Objective

The objective of this work is to develop explanation methods that give pro and con reasons for recommendations, that users can perceive as a fair and transparent approach to explanation generation when compared to current explanation patterns.

Regarding performance, explanation coverage must be above 70%, i.e., explanations must be available for at least 70% of the recommendations presented to users. Moreover, there is a time-constraint for the presentation of explanations, due to the conversational nature of the interaction. Explanations must be given in less than 5 seconds. To understand why we defined these numbers, please refer to Section 4.1.

Achieving these objectives is relevant to the development of conversational agents and RSs with real ability to interact with humans through complex dialogues. The ability to explain decisions is essential in such interactions as interpretability increases user trust and, hence, acceptance of recommendations [46, 48].

## 1.2 Structure of the Work

In Chapter 2, we present theoretical background needed to understand our contributions. In Chapter 3, we present our proposals. In Chapter 4, we put forth the functional and non-functional requirements of our RS, outline its software architecture and detail the generation of explanations. We also describe the algorithms we developed and the training parameters we used for the machine learning models.

We describe experiments, tests and evaluation in Chapter 5. In that chapter, we also analyze and discuss our results. Finally, in Chapter 6 we state the conclusions, contributions and suggest future directions.

## 1.3 Disclaimer of Collaboration

This work has been carried out partly in collaboration with another master's student, Gustavo Padilha Polleti, who successfully defended his work "Explanation Generation For Conversational Recommendation Systems Based On Knowledge Embeddings" [56].

Gustavo developed the explanation algorithms that employ Reasons For and came up with the idea that explanations should present reasons for and reasons against. The author developed algorithms for explanations that contain reasons for and against. The author also implemented and tested them offline. Together, both designed and carried out user experiments to assess these algorithms.

Unless stated otherwise, all other contributions in this work are done by the author.

# 2 BACKGROUND AND RELATED WORK

*"The saddest aspect of life right now is that science gathers knowledge faster than society gathers wisdom."*

-- Isaac Asimov

This chapter provides a comprehensive overview of essential concepts crucial to grasp the underlying principles of this work. We begin by introducing three key concepts: knowledge graphs, recommender systems, and explainable artificial intelligence, elucidating their significance and relevance. Next, we delve into the theory of reasons for and reasons against, establishing the groundwork for the explanation algorithms we have developed.

## 2.1 Recommender Systems

A recommender system is engineered to support decision-making processes. Primitive recommender systems are rankings of best movies, top ten lists or classified offers that can still be seen in magazines and newspapers. However, these rankings or lists do not take user preferences into account. Instead, they are built on the assumption that the viewers of the communication vehicle follow a particular profile. The advent of recommender systems brought the possibility of user-specific rankings that take personal preferences as the central guide in providing suggestions [65].

A recommender system (RS) operates based on a collection of users $\mathcal{U}$ and a collection of items $\mathcal{I}$. By utilizing this data, it generates a score denoted as $r(u, i)$, representing the affinity or compatibility between a particular user $u \in \mathcal{U}$ and an item $i \in \mathcal{I}$ [65]. This score serves as the criterion for ranking a number $N$ of items that will be presented to the user. Ultimately, the user has the autonomy to decide which suggested option aligns best with their preferences or needs.

The definition of the affinity score $r(u, i)$ can vary depending on the specific application domain. Different approaches and techniques are employed to calculate this affinity, taking into account factors such as user behavior, item characteristics, computational performance requirements and other relevant contextual information. At present, the state-of-the-art techniques learn the affinity between users and items from previous experiences using latent variable (also called unobserved variable) models, which often depend on matrix factorization and embedding techniques [28, 30, 33]. These approaches involve a process of transforming item attributes into a latent space. In this latent space, the mapping of similarity is translated into distances, ensuring that similar items are positioned close to each other. Consequently, this arrangement minimizes the distances between related objects. The purpose of this transformation is to create a representation where the proximity of items reflects their similarity [51].

One limitation of latent models is their lack of transparency, which makes it challenging to comprehend the rationale behind the recommendations they provide. This lack of transparency hinders the interpretability of the recommendations, as highlighted by Doshi-Velez and Kim [20].

In this work, we take Miller's intuitive definition of interpretability: the extent to which a human can understand the reasoning behind a specific decision made by the system [46].

Transparency is another crucial concept in this context. While a system may be transparent, allowing users access to its inner workings, it can still produce recommendations with low interpretability. In cases where interpretability is low, one possible approach is to generate explanations for the system's decisions. These explanations serve as a means of providing additional transparency into the system's decision-making process without requiring changes to its internal mechanisms.

A wide range of techniques exist for generating explanations for machine learning models, as detailed in Molnar's Interpretable Machine Learning book [48]. These techniques adopt various approaches to provide insights into model predictions. One approach involves investigating the sensitivity of model outputs to changes to the inputs or the components of the model itself. These types of explanations highlight the evidence of which parts of the input or model contribute to changes in the outcomes. Other techniques focus on more intricate explanations. Some specifically concentrate on certain model types, such as neural networks, and generate explanations by examining the activation or deactivation of specific neurons and layers within the network.

Furthermore, there are explanation generation techniques that are model-agnostic, meaning they are independent of the original model to be explained. These methods only require access to the inputs and outputs of the original model. In this work, explanation-generation methods are of the model-agnostic type. They are designed to provide explanations for any given model, ensuring flexibility and applicability across different model architectures. Note that this does not mean, however, that our methods will work well for any model.

As a further note on the matter of interpretability, it is not sometimes stated that performance and interpretability are conflicting goals [64]. For instance, achieving high accuracy in a classifier often leads to increased complexity, making it more challenging to interpret the underlying decision-making process. However, matters are more delicate in the context of recommender systems, as its the overall performance depends on its interaction with users, not only on traditional machine learning metrics such as accuracy and precision. User trust in the recommendations plays a crucial role in system performance [54], and higher interpretability is strongly linked to higher trust. When interpretation fails, existing recommender systems can exhibit unexpected failures [21].

Previous research has explored various approaches to achieve a balance between performance and interpretability in recommender systems [44, 41, 84]. Some of these efforts involve generating explanations that support the recommendations [49, 3, 31].

To properly evaluate the performance of recommender systems, it is necessary to consider both qualitative and quantitative measures. As mentioned in Chapter 1, in the early stages of this technology, prediction accuracy was primarily used as the evaluation metric. However, a more comprehensive evaluation framework that incorporates interpretability and user satisfaction is essential for a more adequate assessment of recommender system performance [65].

According to [68], accuracy is used due to the assumption that users will prefer a recommender that offers more accurate predictions. How we calculate this prediction accuracy will depend on whether we are measuring rating prediction or usage prediction accuracy [68]. We shall compare the system results to user ratings. Thus, we use one of the prevalent mathematical metrics for error calculation, such as Root Mean Square Error (RMSE) or Mean Absolute Error (MAE). That is because the accuracy itself is a measure of the difference between the observed and the actual value of a quantity, the so-called observational error [10]. For the calculation of accuracy according to RMSE and MAE, let us consider a recommender has access to a test set $T$ and predicts ratings $\hat{r}_{ui}$ for this set. We represent user-item pairs by $(u, i)$, the actual ratings $r_{ui}$ are given, and then, the equations are as follows:

Table 1: Confusion matrix with recommendations and usage [65].

| | Recommended | Not recommended |
|---|---|---|
| Used | True-Positive (TP) | False-Negative (FN) |
| Not used | False-Positive (FP) | True-Negative (TN) |

$$RMSE = \sqrt{\frac{1}{|T|} \sum_{(u,i)\in T} (\hat{r}u_i - r_{ui})^2} \quad , \tag{2.1}$$

$$MAE = \sqrt{\frac{1}{|T|} \sum_{(u,i)\in T} |(\hat{r}u_i - r_{ui})|} \quad . \tag{2.2}$$

Currently, accuracy is no longer the only evaluation metric, as more complex requirements have been used, such as fairness, trust and transparency [20]. However, prediction accuracy is still primarily used for experiments not involving users, the so-called *offline* evaluation.

When one is able to test the system with users, one objective metric of interest is usage prediction accuracy. That is a measure of how useful the suggestions were to the users. To calculate prediction accuracy, it is useful to apply a confusion matrix as seen in Table 1. One may calculate not only prediction accuracy, but also precision and recall as follows [65]:

$$\begin{aligned} accuracy &= \frac{TP + TN}{TP + FP + TN + FN}, \\ precision &= \frac{TP}{TP + FP}, \\ recall &= \frac{TP}{TP + FN}. \end{aligned} \tag{2.3}$$

While it may not always be feasible due to various constraints, the primary approach for evaluating recommender systems is through tests with real users. This user-centric evaluation allows for more specific and clarifying questions to be asked, providing valuable insights into user preferences, satisfaction, and overall system performance. Additionally, the metrics discussed in this section can serve as supplementary evaluation criteria to assess the system's effectiveness.

By involving real users in the evaluation process, researchers and developers can gather direct feedback, understand user behavior, and make improvements based on user needs. This approach goes beyond solely relying on quantitative metrics and adds a qualitative dimension to the evaluation. While practical limitations may restrict the extent of user testing, incorporating real users into the evaluation process enhances the validity and reliability of the results, leading to more meaningful and user-centric recommendations.

## 2.2 Knowledge Representation

There are several ways to represent the knowledge an agent has at its disposal. Example representations are logical sentences and knowledge graphs. The former uses predicate logic while the latter relies on entities and relations [67, 43]. We employ knowledge graphs in this work, and we describe them in further detail next.

### 2.2.1 Knowledge Graph

A graph $G$ is a pair $(V, E)$ where [29]:

- $V$ is a finite set whose elements are called the vertices of $G$;
- $E$ is a set of unordered pairs $v, w$ where $v, w \in V$ and $v \neq w$. The elements of E are called the edges of G.

A knowledge graph is a graph whose vertices, also called nodes, are entities, and edges represent relations between those entities. A triple $\langle$*head, relation, tail*$\rangle$ is called a fact, and the knowledge graph as a whole, i.e. the set of all facts, is a knowledge base [32].

The knowledge base used in this work, USPedia[1], represents knowledge about courses from the University of São Paulo (USP). An example fact in this base would be the triple $\langle$*machine_learning, subject, PCS3838*$\rangle$, which means machine learning is a subject taught in the course PCS3838 or that this unit's content include machine learning.

Another example knowledge graph is the one shown in Figure 1. Those are parts of a knowledge graph, generated by a query using a graph database called Neo4j. Movielens is a set of data-sets for movie recommendations from the research group GroupLens from the University of Minnesota, widely used for research purposes [34]. In the image, one sees a set of users (3, 4, 13, 18, 19, 21, 20) who have the action genre as a favorite. We then see movies related to "Waiting to Exhale (1995)" and a number of movies whose genres include action or, put another way, a number of action movies.

[1]The name USPedia comes from *USP*, as the University of São Paulo known, and the suffix *Pedia* as in WikiPedia or, more appropriately in our case, DBPedia. Although the knowledge base was first meant to be about all courses from the university, it ended up having only facts about courses from the Polytechnic School. The name was maintained nonetheless and some day the base might be expanded.

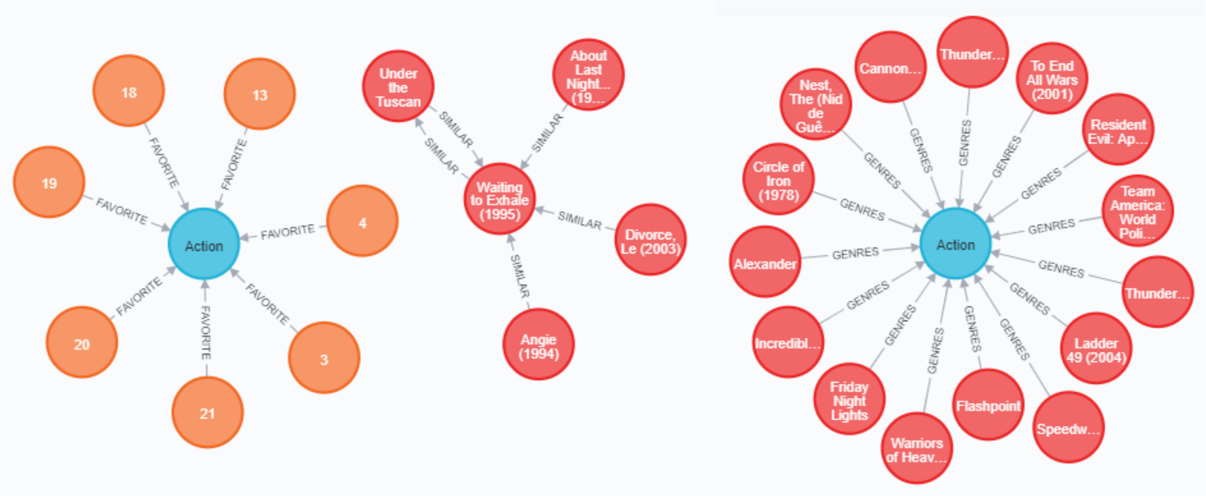



Figure 1: Example graphs from MovieLens. Source: Author.

### 2.2.2 Knowledge Graph Embeddings

A knowledge graph contains mostly symbolic, textual data which would be hard to manipulate in tasks such as prediction of relations. A solution is to produce embeddings from these knowledge graphs. An embedding maps entities and relations from a knowledge graph into a latent representation in continuous vector spaces. This representation has advantages in various tasks such as knowledge graph completion and entity classification [77].

Embeddings are usually built in a three-step process. First, entities and relations are represented numerically in a continuous vector space. Next, a scoring function $f_r(h,t)$ is defined on each triple $\langle h,r,t \rangle$ so as to indicate the plausibility that the triple holds. The third and final step is to learn the representation of the entities and relations, i.e., the embedding itself. That is accomplished by optimizing the plausibility of all triples (facts) in the knowledge graph [51, 77].

A particular embedding model is TransE, which we used in this work. TransE (from "Translational Embedding") represents entities and relations in the same space, as vectors $\mathbf{h}$ and $\mathbf{t}$. The relation is then a vector $\mathbf{r}$ that transforms through space translation such that $\mathbf{h}+\mathbf{r} \approx \mathbf{t}$ when the fact $\langle h,r,t \rangle$ is true [11, 77]. Figure 2 illustrates the idea; the scoring function for TransE is expressed in Equation (2.4). The higher this scoring function, the higher the plausibility of the triple $\langle h,r,t \rangle$:

$$f_r(h,t) = -\|\mathbf{h} + \mathbf{r} - \mathbf{t}\|_{\frac{1}{2}}. \tag{2.4}$$

Note that the $\frac{1}{2}$ term in Equation (2.4) refers to the L2-norm, also referred to as

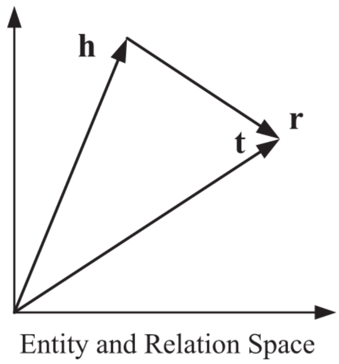



Figure 2: Representation of facts through TransE. Source: [77].

the Euclidean norm. Further details can be found in the survey [77] and in the original TransE paper [11].

In this work, we use the terms knowledge graph embedding, embedding of knowledge graphs and knowledge embedding as synonyms. For conciseness, we also use the acronym KE, from knowledge embedding.

## 2.3 Explaining Recommendations

The section introduces concepts and techniques that generate explanations in the context of recommender systems. The section examines both model-specific and model-agnostic techniques for generating explanations, highlights the impact of explanations on user satisfaction and decision-making and its role in improving transparency.

Explanation generation belongs to a field referred to as Explainable Artificial Intelligence (XAI), a term coined by the Defense Advanced Research Project Agency (DARPA) in 2016. That agency funded researchers to foster the development of interpretable and explainable artificial intelligence systems so as to cope with the technological and ethical challenges posed by the advancement of highly accurate but hardly auditable systems. These are also called black-box models as they are metaphorically seen as opaque, making it difficult to see how they give their results [26].

There are machine learning models which are interpretable by construction, such as decision trees and linear regression. These models require no further explanation since, for example, given the equation a linear regression outputs, one can see the effect each weight has on the outcome. For instance, consider the three-dimensional equation $y = \beta_0 + \beta_1 x_1 + \beta_2 x_2$. $\beta_1$ represents how much $x_1$ contributes to the result $y$ and $\beta_2$ how $x_2$ contributes, while $\beta_0$ is what one would have without contribution from neither $x_1$ nor

$x_2$.

There are cases where interpretable models are not good enough in terms of prediction accuracy or other metrics. In these cases, other more flexible models such as deep neural networks are necessary. These models, however, must be explained [48].

Algorithms for explaining black-box models can be categorized as model-specific or model-agnostic. Model-specific algorithms are designed to work exclusively with a particular algorithm. In contrast, model-agnostic algorithms operate based on the input and output of the model, independent of its specific architecture. They can be either local, focusing on explaining individual instances or a subset of the data, or global, providing explanations for the entire output data set based on the overall input data set [48].

An example of local, interpretable method for generating explanations is LIME - Local Interpretable Model-agnostic Explanations [63]. Intuitively, LIME works by using an interpretable model to capture the function [1] learned by the black-box for a subset of the data points to be predicted. One may ask: how can an intrinsically interpretable, thus less flexible model, learn the same function a a complex black-box model? The answer is that the interpretable model only learns the function for localized data points.

A surrogate model is an interpretable model that approximates the predictions of a black-box model either locally (Local Surrogate) or globally (Global Surrogate). A surrogate model often generates explanation for instance $x$ as a model $g$ that minimizes loss function $L$ while maintaining complexity $\Omega$ low [48]:

$$\arg\min_{g} L(f, g, \pi(x)) + \Omega(g), \tag{2.5}$$

where $\pi(x)$ defines a region around $x$.

Figure 3 depicts a toy example from Ribeiro et al. original paper [63]. In the image we have a bright red cross that is the instance of interest. The other red crosses and blue dots are perturbed samples and the blue and red colors mean that these points belong to the blue of red class. These classes are divided by a complete decision boundary; LIME finds a line that represents the decision boundary for the sample points locally. We can see that the approximation does represent the pattern and divides the points with high accuracy in this example.

In addition to the efforts of the scientific community to provide explanations for data processing by artificially intelligent mechanisms, there is also a legal obligation brought by the European General Data Protection Law (GDPR) [25]. In its Article 22 - Auto-

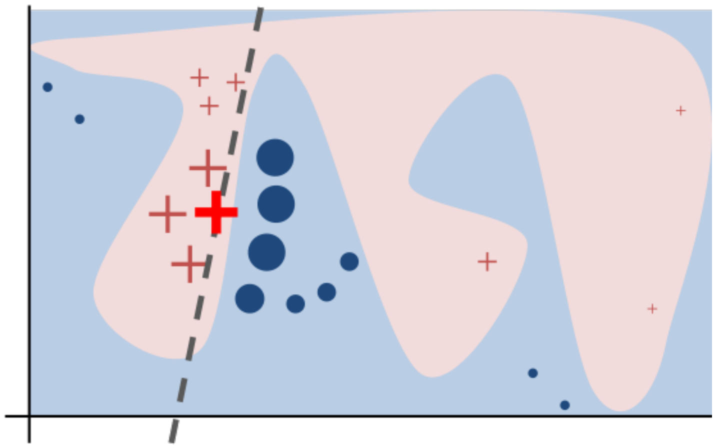


Figure 3: Intuitive example of explanation for the bright red cross.

mated Individual Decision-making, Including Profiling - it reads [55]: "The data subject shall have the right not to be subject to a decision based solely on automated processing, including profiling, which produces legal effects concerning him or her or similarly significantly affects him or her." The article requires that users be granted access to an explanation for results from automated data processing. For the technical personnel responsible for giving the requested explanations themselves, it is thus paramount that the algorithms are either intrinsically interpretable or that explanations are generated [25]. As many high-performing algorithms are black-boxes [2], this has created a demand for explanation generation.

## 2.4 Explanations and Knowledge Graph Embeddings

In this section we describe the type of Conversational Recommender System (CRS) we deal with in this work, review a method for generating explanations from a knowledge graph and techniques to improve coverage through knowledge graph embeddings [59]. This sets the stage for our proposal, described in Chapter 3.

First, suppose the CRS receives a query from the user. This query asks for a topic of interest that is represented as an entity $e_h$ in an auxiliary knowledge graph. The CRS must then find items (entities) that are linked to the entity $e_h$ though some relation $r$ that represents the affinity between the user preference and the items. That is, the system must find triples following the pattern $\langle e_h, r, e_t \rangle$ for possibly many tail entities $e_t$. To do so, the CRS must predict links connecting various entities; we suppose that link prediction operates using a selected knowledge embedding (KE). That is, the CRS returns the top $N$ ranked entities, where $N$ is an hyper-parameter fixed for the CRS, and the ranking follows the plausibility score $f_r(e_h, e_t \mid \Theta)$ specified through the embedding. Here the score given by the embedding is a predicted measure of how strong the item entities relate to the user preference. Parameters $\Theta$ are tied to the embedding and are learned from previously processed training data.

To illustrate, take a system that recommends classes to students based on a given topic of interest. Suppose a student sends the query "recommend me something about Astronomy" to which the CRS might produce the output "I recommend you Exoplanets101". The CRS presumably identified Astronomy as the topic (item) of interest, and found the possible classes to recommend by accessing a previously computed list $\mathbb{T}$ containing the plausibility of a link to each entity in its auxiliary knowledge graph through the relation subject. Possibly the CRS has several entities at its disposal in descending order of plausibility, from Exoplanets101 to say Mechanics101, Mechanics202, and so on. In our example, we have

$$\mathbb{T} = \begin{bmatrix} f_{\mathsf{subject}}(\mathsf{Astronomy}, \mathsf{Exoplanets101} | \Theta) \\ \vdots \\ f_{\mathsf{subject}}(\mathsf{Astronomy}, \mathsf{Mechanics202} | \Theta) \end{bmatrix}.$$

Note that this conceptual model of the CRS is agnostic to the specific embedding and plausibility function, as it only assumes the existence of a plausibility function. In short, the CRS recommends the entities that best fill the question mark in the query $\langle e_h, r, ? \rangle$, where $r$ is a relation modeling how tail entities meet user preferences $e_h$.

Given this description of the recommendation scheme, we move now to describe explanation methods from that use the same knowledge base as the recommender to generate explanations. The first method we present focuses on generating explanations using the knowledge embedding that are faithful to the original knowledge graph [59, 56]. The second method improves on the first, by reducing the execution time and improving coverage [56, 60].

The first method is a local surrogate one [59], inspired by LIME, that employs an intrinsically interpretable machine learning model to explain local instances of black-box predictions reached through embeddings of knowledge graphs. As local surrogate explanations usually are weighted features and, in the case of embeddings, features are not interpretable, a suitable interpretable representation for the embedding is proposed. Then a method that extracts those representations, or maps embeddings to interpretable representations, is proposed. The idea is to use a graph feature model, with one-sided path features to represent the embeddings as information that users can grasp. To extract these features, note that embeddings allows one to find plausible facts in the original graph. For instance, consider the following example from the original work [59]:

**Example 1.** *Here we show a toy example where we are interested in the tail prediction for the triple $\langle astronomy, topic\_of\_class, ?\rangle$. Let us define:*

$$\mathbb{T} = [f_{topic}(e_{astro}, e_i \mid \Theta), e_i \in \mathcal{E}],$$

$$sort\_desc(\mathbb{T}) = \begin{bmatrix} f(e_{astro}, e_{exoplanets101}|\Theta) \\ f(e_{astro}, e_{aeronautics102}|\Theta) \\ \vdots \\ f(e_{astro}, e_m|\Theta) \end{bmatrix}, \quad g_{topic}(astro|\Theta) = exoplanets101. \quad (2.6)$$

*The list $\mathbb{T}$ represents the plausibility score calculated for all entities. Thus, to discover the most plausible candidates for courses about astronomy, we sort $\mathbb{T}$ in descending order and identify that the class most presumably related is exoplanets101, then aeronautics102 and so on. Our function $g_r$ returns only the top 1 ranked entity, in this case exoplanets101.*

*Consider also the embedding feature extraction for the compound feature category − topic of the entity astronomy. First, we discover the category of astronomy using the function $g_{category}$, then we inquire for her classes using $g_{topic}$. That is,*

$$g_{category}(astro|\Theta) = astrophysics \rightarrow g_{topic}(astrophys|\Theta) = mechanics101. \quad (2.7)$$

The function $g_{category}(astro|\Theta)$ is in fact a classifier provided by the embedding. For the top-1 tail prediction task, the embedding can be viewed a set of black-box classifiers $g_r \in \mathcal{G}$, for each relation $r \in \mathcal{R}$, where $g_r(h|\Theta)$ returns the tail entity $e_t \in \mathcal{E}$ that yields the highest plausibility score $f_r$ for the triple $\langle e_h, r, e_t\rangle$. In more formal terms:

$$g_r(e_h|\Theta) = \arg\max_{e_i \in \mathcal{E}} f_r(e_h, e_i \mid \Theta). \quad (2.8)$$

As there is a real-valued vector representation for each entity $e_i \in \mathcal{E}$ in the KE parameters $\Theta$, the authors [59] define the classifier function $g_r$ such that it takes the head entity embedding as input. Having completed this mapping and generated the interpretable representation for the embedding, the next step is to general the explanations. For that, they follow an approach similar to LIME [63], but in this case, one is to sample around the original input representation. To explain why an entity $e_t$ is a plausible tail entity for a triple $\langle e_h, r, ?\rangle$, one first samples $K$ data points around $\mathbf{e}_h$, thus generating a data-set

**Algorithm 1** Local Surrogate Explanation Generation. Source: [59].

**procedure** EXTRACT-FEATURES($\hat{z}_k, \Pi_L, \mathcal{G}$)<br>
  $\phi_k = \{\}$<br>
  **for all** $\pi \in \Pi_L$ **do**<br>
    $e_\pi \leftarrow \hat{z}_k$<br>
    **for each edge** $r_j \in \pi$ **do**<br>
      $e_\pi \leftarrow g_{r_j}(e_\pi)$ ▷ Plausibility-based classifier for relation $r_j$<br>
    **end for**<br>
    $\phi_k \leftarrow \phi_k \cup e_\pi$<br>
  **end for**<br>
  **return** $\phi_k$<br>
**end procedure**<br>
**procedure** EXPLAIN-INSTANCE($\hat{e}_h, r, t, L, \mathcal{G}$)<br>
  $\Phi \leftarrow \{\}$<br>
  $\Pi_L \leftarrow$ GRAPH-FEATURES($L$) ▷ Generate set of path features<br>
  **for** $k \in 1, 2, 3, ..., K$ **do**<br>
    $\hat{z}_k \leftarrow$ SAMPLE-AROUND($\hat{e}_h$) ▷ Generate perturbed sample<br>
    $\phi_k \leftarrow$ EXTRACT-FEATURES($\hat{z}_k, \Pi_L, \mathcal{G}$)<br>
    $\Phi \leftarrow \Phi \cup \langle \phi_k, g_r(\hat{z}_k), d(\hat{e}_h, \hat{z}_k) \rangle$<br>
  **end for**<br>
  $g'_r \leftarrow SLR(\Phi)$ ▷ Train interpretable classifier with $\phi_k$ as features and $t$ as target<br>
  Draw explanations from $g'_r$ in terms of feature importance<br>
**end procedure**

$\mathcal{Z}$ of perturbed samples $\hat{\mathbf{z}}_k$. Next, for each perturbed sample $\hat{\mathbf{z}}_k \in \mathcal{Z}$, one must carry out the feature extraction presented above. We get the interpretable representation:

$$\phi_k = [e_\pi : g_\pi(\hat{\mathbf{z}}_k), \pi \in \Pi_L]. \tag{2.9}$$

At this point we have the interpretable representation $\phi_k \in \Phi$ for each perturbed sample $z \in \mathcal{Z}$, so one may train an intrinsically interpretable classifier $g'_r \leftarrow SLR(\Phi)$ and use feature importance as explanations.

We present the pseudo-code for this method in Algorithm 1, as described in the original paper [59]. This particular method is able to produce explanations that are faithful to the original graph, but it comes with high computational cost. This happens due to the need to train a new interpretable model for each new instance to be explained. Another relevant previous work simplifies and improves the method we just described, as we now discuss.

In the paper "Explanations within Conversational Recommendation Systems: Improving Coverage through Knowledge Graph Embeddings" [60], the authors define an *explanation* as a path of length at most $L$, where $L$ is a hyper-parameter, starting and

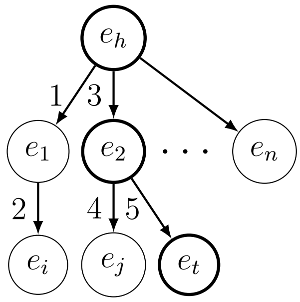



Figure 4: Depth-First Search toy example. Source: [60].

ending at selected entities $e_h$ and $e_t$. For instance, we may want to find the explanation for a case where the user wishes to learn about Astronomy and receives Exoplanets101. We may then return the following part of the knowledge graph:

$$\mathsf{Astronomy} \xrightarrow{\mathsf{subject}} \mathsf{Exoplanets} \xrightarrow{\mathsf{subject}} \mathsf{Exoplanets101}.$$

To reduce computational cost and execution time, the authors propose to generate explanations by first selecting a set of paths $\Pi_L$ of length at most $L$; for instance, one may focus on paths that have sequences of links labeled with relation subject, or perhaps by either subject or topic_of. They assume the construction of $\Pi_L$ is domain-dependent and that, although one may use all possible paths of length at most $L$, this may be unfeasible in practice. They also note that if one selects few paths, good explanations may be unavailable. They assume that a professional building a CRS will be able to impose sensible restrictions on possible paths (or perhaps consult a domain expert who will do so). For large knowledge graphs, they suggest automated methods such as graph feature selection algorithms [24] may be needed.

To generate an explanation according to this method, the search should run through every path $\pi \in \Pi_L$ starting from $e_h$, using a depth-first search (DFS). If the search reaches $e_t$, the path from $e_h$ to $e_t$ is a possible explanation. Note that the search-tree height is at most $L$; that is why they use DFS instead of, for example, breadth-first search.

Running the DFS using solely the links in the knowledge graph has the drawback of *low coverage* of explanations. This happens because most knowledge graphs are sparse and they found the same in preliminary testing. To be able to properly explain recommendations to users, one would like to have a hundred percent coverage, covering every recommendation with an explanation. One possible strategy to increase coverage is by going beyond the edges in the original knowledge graph; they do so by running the search in an enlarged (weighted) graph [60].

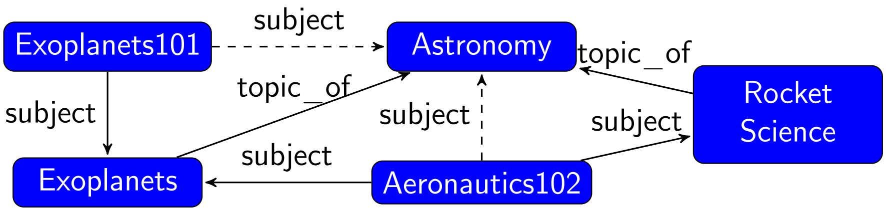



Figure 5: Sub-graph illustrating explanations for the recommending Exoplanets101 and Aeronautics102. For example, the path Exoplanets101 $\xrightarrow{subject}$ Exoplanets $\xrightarrow{topic\_of}$ Astronomy explains the recommendation Exoplanets101. Source: adapted from [56].

That is, we wish to run the search in the space containing completions of the knowledge graph as established by the knowledge embedding. However, the embedding is a continuous mapping, not a graph. How can we search on it? The authors use the KE itself to extract interpretable features. But, instead of using one-sided paths as features, they use paths formed by a number $l$ of arbitrary relations in the form $r_1 \rightarrow r_2 \rightarrow ... \rightarrow r_l$ [60]. The difference between one-side path and this last format, that is similar to PRA [38], is that one-sided paths are less restrictive and are used for correlation while PRA-styled ones allow for inference.

To summarize, an explanation is taken to be a sequence of relations $\pi = \{r_1, r_2, \ldots, r_L\}$, that is built only if we can find triples from $e_h$ to $e_t$ such that each triple $\langle e_i, r_k, e_j \rangle$ either is in the original knowledge base, or it displays a plausibility score such that $f_{r_k}(e_i, e_j|\Theta) > \delta_{r_k}$. The search starts from $e_h$, moving downward in the search tree, according to the plausibility scores, to another entity. Even though the search will stop at some point, it makes sense to stop after some desired time limit. In our own scenario, in a conversational setting, it is natural to set a time limit for the user to receive a response.

To illustrate procedure above, consider the example of a DFS in Figure 4. The nodes are sorted from the highest plausibility score on the left to the lowest on the right. The numbering on the edges indicates the order in which each node is visited. Please also consider the toy example in Figure 5; it indicates that Exoplanets101 is recommended because it is about Exoplanets, which is a topic of Astronomy. The graph also contains the explanation for recommending Aeronautics102 because it teaches Rocket Science, which is also a topic of Astronomy. Note that the student wishes to learn about Astronomy in this case.

To conclude this section, we summarize the two methods we described. In their first method it was necessary to train a surrogate model and to compute feature importance so as to know which features could explain the knowledge embedding prediction [59]. In the second method [60], the set of key features $\Pi_L$ is given by the domain expert and so as to avoid re-training the surrogate model for every instance one wishes to explain.

## 2.5 Bipolar Argumentation

Computational argumentation is a sub-field of computer science that deals with argumentation supported by computers, be it monological or dialogical. A monological argument occurs when a single communicating agent, or interlocutor, is acting. A dialogical argumentation, on the other hand, happens between two interlocutors at a time. According to [14], computational argumentation follows three steps:

1. the exchange of arguments, whereby communicating agents (interlocutor) transmit an argument, i.e. an explanation or justification;
2. the valuation of interacting arguments, when the receiving interlocutor assigns weights to each argument received;
3. selecting the most acceptable argument, the end result of the argumentation.

Arguments tend to defend a certain position. In a bipolar argumentation framework, arguments are given in a bipolar (positive/negative) manner. In other words, arguments are given in favor and against a certain position [14]. That is similar to the reasons for and against discussed in the previous section. Bipolar frameworks can enable new types of interactions between arguments as well as new kinds of valuations for the arguments. Results of argumentation processes may be presented as acceptable and rejected arguments [5].

These ideas in computational argumentation may be applied to explanations that rely on reasons for and reasons against an issue.

## 2.6 Critiquing-based Recommenders

A critiquing recommender system is defined by its dependence on user feedback, also called review or critique [27]. Through such critiques one may compare a number of options according to their score on specific characteristics.

For instance, let us imagine Anna wants to buy a mobile phone. This phone may be evaluated according to the quality of its camera, its battery and its operating systems.

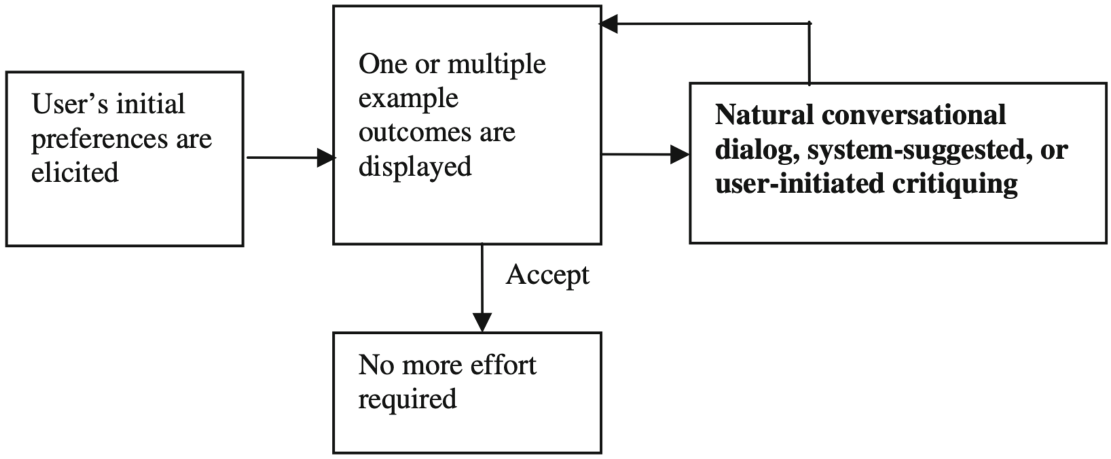



Figure 6: Architecture of a critiquing recommender system (note the feedback loop). Source: [15].

If other users rated their characteristics previously or these are even given by experts, Anna could use these very characteristics to assist her decision on whether to follow the recommendation or not. A critiquing capability is then added to allow her to enter which of those characteristics could be changed in order to find a more suitable phone for her [16]. As an example, she could want a phone with a battery of longer time, and the system would search and possibly fetch a similar phone with an improved battery time.

The overall workings of a critiquing RS are seen in Figure 6 [15]. The figure shows the architecture diagram with the feedback in boldface to highlight it as the main feature of such systems.

Researchers have developed systems that can proactively suggest critiques for the user to follow. For example, give the user the choices of "lower price" and "'smaller" to a given phone [15].

In another proposal, characteristics of a product are presented relative to an individual item; this is called a compound critique [81]. Such reasons for and against are produced by the so-called Multi-Attribute Utility Theory (MAUT). This theory accounts for the conflicting value preferences for item attributes and produces a score for each item that gives the degree to which it satisfies user preferences [36].

One can take such strategies as explanations that indicate how a product compares to others, by showing reasons that either support or discredit buying the product. The system developed by [81] outputs textual explanations of the form "More Memory, Larger Hard-Disk, Lighter and Cheaper. But Different Type of CPU, slower CPU, Smaller Screen and Shorter Battery Life". Such an explanation contains the reasons supporting

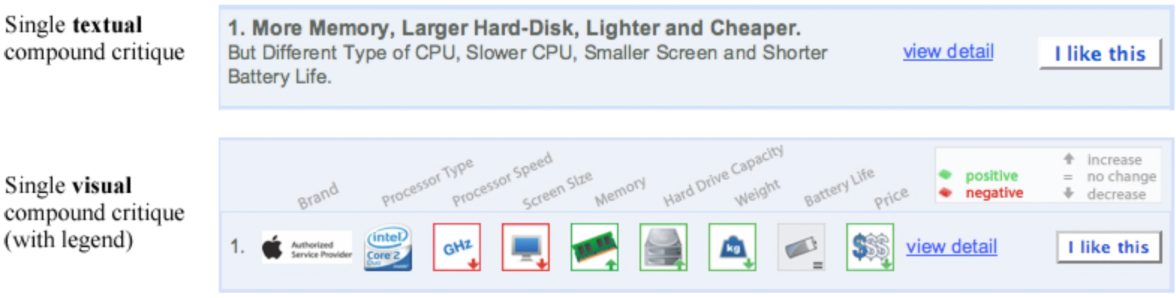



Figure 7: Example interfaces with the comparison of attributes. Source: [80]

the product in the first sentence and the reasons against the purchase in the second sentence.

In another work, authors have adopted a graphical interface instead of text, using colors to indicate whether the attribute had a *pro* or *con* effect on the item. Red represented a negative effect and green a positive one [80]. An illustration of these two presentation schemes can be seen in Figure 7.

## 2.7 A Theory of Reasons

In Section 2.4 we saw a method for generating explanations for the education domain using paths in a knowledge graph. These explanations consist of reasons that support the recommendations given by the system. In [60], we demonstrated that providing those reasons has effects on user experience. Users are to understand the recommendations and to reach increased trust in the system due to the explanations.

However, one could argue that explanations may be misleading to the user by giving extra facts to support a recommendation. A possible real-life case with comparable conflict of interests is as follows.

Let us assume a potential customer (Anna) walks into a coffee shop. The customer is looking for a gift for a friend (Belle) who is having her birthday soon and loves coffee. At this point, Anna is getting some help from the saleswoman at the shop - Clair. Clair then suggests, among a relatively large selection of coffee, that Anna takes one or both of two coffee beans which are the most expensive in the shop. As the prices are above Anna's expectations, she asks what makes those kinds of coffee so appealing. Clair then replies that they are of the highest quality, produced by cooperative small-farmers in Ghana, and satisfy Belle's taste.

One could ask at this point: is Clair fair? Is there a way her explanations could be *better*? The question should be *better for whom*? If they mean better for Anna, then we might say that there are probably missing pieces of information that could be helpful for Anna to make a more informed decision. If we give her these missing pieces of information, we provide her with ways to decide which options are better for her according to her own judgment.

Let us consider what other researchers have done in this direction.

In Philosophy, we find the work of Justin Snedegar, who gives a comprehensive discussion on the topic, in particular on contrastive normative reasons [70]. Contrastive means that reasons for or against a certain option are given compared to other options. For instance, that a fact $b$ supports a hypothesis $h$ rather than an alternative hypothesis $\eta$, not that $b$ supports $h$ *simpliciter*. These reasons are also called *justifying* reasons because they justify some action [4].

In another work [71], Snedegar points out that in general, it is believed that reasons against some option are just reasons for some competing options, but proceeds to affirm that reasons against and reasons for are two different entities. He proposes a way of viewing those reasons that satisfy his claim.

### 2.7.1 Simple View of Reasons For and Against

Snedegar presents four versions of what he calls the *simple view* of reasons for/against: what you should do is determined by a competition between reasons that support the available options. This view, he adds, is often depicted by a metaphor of a scale with pans. In this case, each pan represents an option, and each marble on the pans represent reasons weighing heavier or lighter for the option. A critique of this metaphor is that it is incomplete because, for example, it assumes that multiple reasons for an option can be combined and that the total weight of the combination would equal that of the sums of the individual weights [71]. That is not necessarily true as there might be some sort of intersection among those reasons.

Although the *simple view* has its limitations, Snedegar affirms that it is attractive due to its simplicity. It is simple because it considers unique relations between reasons and options - the *for* relation - and also a unique type of competition - that between reasons that support conflicting options [71]. That simplicity goes well with the scientific community's preference for simple over complex theories [53, 76].

In broader terms, the *simple view* states that a reason against an option P is a reason for **alternatives to P**. Snedegar breaks it down to what he calls four different implementations, depending on these alternatives. In the next paragraphs, we will go through each of these implementations.

Let us first define some terms. The set of possible options on is $O = \{P, Q, R, S\}$. The set of reasons for P is denoted by $\mathcal{R}_{for}(P)$, for Q by $\mathcal{R}_{for}(Q)$ and so for R and S. The set of reasons against P is named $\mathcal{R}_{ag}(P)$ and so on.

The first implementation takes the reasons against P to be reasons for some alternative to P. For instance, the reasons against P are the union of reasons for Q, reasons for R and reasons for S: $\mathcal{R}_{ag}(P) = \bigcup \mathcal{R}_{for}(r \in O \backslash P) = \{q1, q2, q3, r1, r2, r3, s1, s2, s3\}$. This implementation has the drawback of considering that there are reasons against P which are also against Q, $\mathcal{R}_{ag}(P) \cap \mathcal{R}_{ag}(Q) = \{r1, r2, r3, s1, s2, s3\}$. But in practice, that is hardly the case, and Snedegar gives the example of a restaurant with long waiting times as a reason against going there. That is a reason for going to another restaurant, but there could be another restaurant among the options that also has long waiting times. The implementation in question fails to represent that fact since it does not take into account that reasons against this second restaurant could also be reasons against the first one.

In the second implementation, a reason against P is a reason for not doing P, i.e., a reason for the negation of P. Although this solves the problem mentioned in the previous implementation, this one fails to help decide on which of the options to take in computational terms (that is, as a clear and objective algorithm). It does not clarify how reasons that are going against P act on each of the alternatives to P.

The third implementation's reasons against P are reasons for the disjunction of all of the alternatives to P. In this case, $\mathcal{R}_{ag}(P) = \mathcal{R}_{for}(Q \vee R \vee S)$.

The fourth and last implementation in the *simple view* considers reasons against P to be reasons that are a reason for **each** of the alternatives. What this means is that reasons against P are reasons that support all of the alternatives to it. That happens if, in our notation and example, the $\mathcal{R}_{for}(Q) \wedge \mathcal{R}_{for}(R) \wedge \mathcal{R}_{for}(S) \neq \emptyset$ holds true. Going back to the restaurant example, a reason against P satisfying this implementation could be that restaurants Q, R and S accept credit card payments while P does not. The argument Snedegar uses to reject this version of the *simple view* is that, although it does not suffer from the shortcomings of the others, it could happen that a reason against P is not a reason for one of its alternatives. For instance, he states that P being crowded is a reason against going there and, by this version, it is a reason for going to Q and a reason for going to R and also a reason for going to S. However, it happens that S is even more crowded than P and so he rejects all versions of the simple view in favor of his proposal, which solves all of the problems mentioned [71]. He thus looks at a more complex scheme as described in the next section.

### 2.7.2 Reasons For as distinct from Reasons Against

The fundamental claim by Snedegar is that reasons for and reasons against are distinct types of reasons. To make the distinction clear, he uses the idea of objectives. The definition provided in [71] is:

> **For**: r is a reason for A when r explains (or is part of the explanation) why A promotes/respects some objective better than all of the alternatives.
>
> **Against**: r is a reason against A when r explains (or is part of the explanation) why A promotes/respects some objective less well than some alternative.

He explains that objectives are either desires or values of the agent that ought to take action. As desires, objectives could be to feed oneself, to enjoy, to learn something. As values, objectives could be justice, friendship or happiness. So, considering these definitions and the proposal, objectives are what ultimately explain or provide the reasons [71].

Going back to the examples, a possible objective could be that a student wants to have lunch quickly and go back to the library to work on his master's thesis. In this case, if we say restaurant P is the cheapest, that is neither a reason for nor a reason against the student going for lunch there. Why? Although, in general, students have the objective to save, the goal defined here was solely to have a quick lunch. Therefore, P being cheap does not promote the objective better or less well than its alternatives.

We could list some more characteristics of restaurants P, Q, R and S to compare them. Nevertheless, to decide according to Snedegar's proposed framework, the student must know which one has a higher waiting time, which is the farthest from the library, which food takes longer to eat, which could require a longer pause due to a higher *post-lunch slump* and other reasons that relate to the objective of spending less time for the lunch break.

# 3 GENERATING EXPLANATIONS

*"The best way to predict the future is to invent it."*

-- Alan Kay

This chapter describes our contributions by first introducing the recommendation model we assume in a concrete setting, so as to clarify its practical aspects. We then present techniques for generating explanations for recommendations, and then describe our main contributions.

## 3.1 Recommendation Scheme

To examine our proposals in a realistic setting, we developed a recommender system (RS) with a knowledge base (KB) that contains classes from the Escola Politécnica at the Universidade de São Paulo. This KB has entities representing courses, classes, professors, topics, and their relationships. Its primary goal is to assist students in deciding which classes to take next, especially optional ones. For that purpose, students ask the RS questions through textual input (or voice command, that is processed first and then input as text).

From the question, the system infers the users preferences. As the system relies on a knowledge base (KB), there must be an entity $e$ in the KB that is a topic related to the user preference. The task is to find classes that are related to this topic. As classes are also entities, let us differentiate the topic $e_h$ from the class $e_t$. By saying that two entities are related, we mean that there exists a relation $r$ such that the triple $\langle e_h, r, e_t \rangle$ is plausible, according to the knowledge graph embedding prediction. For instance, in the case of the embedding model TransE as pictured in Figure 2, it should be possible to go from $e_h$ to $e_t$ through a translation by $r$.

In practice, several tail entities - classes, in this case - may be related to the head entity of interest - the topics. Therefore, according to the plausibility score of the adopted embedding, the recommender returns the top-N classes that are most closely related to the topic of interest. Recall that the plausibility score is a function output by the embedding $f_r(e_h, e_t \mid \Theta)$, where $\Theta$ denotes a set of hyperparameters.

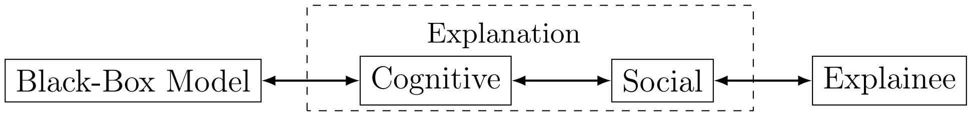



Figure 8: Visual representation of the explanation processes.

Suppose the RS recommends *Digital Systems Design* to a student that asks for *hardware development*. This means that the knowledge graph embedding built a vector of plausibility scores $\mathbb{T}_h$ from $e_h$ (*hardware development*) to the entire set of entities in the embedding, except $e_h$ itself, $\mathcal{E}$. Then, the program sorted $\mathbb{T}_h$ in descending order, and the score of *Digital Systems Design* was in the first position. In a more intuitive form, we may say that according to the classes available in the KB, the data we have about these classes and the embedding model we trained, $e_t$ = Digital Systems Design was the one most closely related to what the student aspires to learn. After we sort them, $\mathbb{T}_h$ and the recommendations $R_h$ may look as follows:

$$\mathbb{T}_h = \begin{bmatrix} f_h(e_h, e_1 \mid \Theta) \\ \dots \\ f_h(e_h, e_n \mid \Theta) \end{bmatrix}, \qquad R_h = \begin{bmatrix} \textit{Digital Systems Design} \\ \dots \\ \dots \end{bmatrix}. \tag{3.1}$$

The system then presents the recommendation "I recommend Digital Systems design, which teaches about hardware development".

As our techniques deal with automatically-generated explanations for the suggestions the RS provides, we now introduce the explanation mechanism.

## 3.2 Explanation Algorithms

The act of explaining requires a social interaction between at least two agents: the explainer and the explainee. The explanation model proposed by Miller defines the explanation as a compound process with two steps: the cognitive and the social. While the first describes the process of identifying the causes as to why a given decision was made, the second is the process of conveying or communicating such reasons to the explainee. Figure 8 depicts this process, that starts with the opaque black-box model, from which we find the explanations comprising the cognitive and social phases; then the explanation is given to the inquirer, the explainee.

This main contribution here is the concrete discussion of reasons against, as those add to the already usual capability to provide reasons for [59, 60]. Reasons against are mostly

**Algorithm 2** Explanation Generation algorithm Reasons-For. Source: [61].

**procedure** REASONS-FOR($i$: item, $u$: user, $\Pi$: set of paths, $\mathcal{G}$: graph)
    $\Phi_{u,i} = \{\}$ ▷ Set of reasons for $i$
    **for all** $\pi \in \Pi$ **do**
        $\phi \leftarrow \gamma(u, i, \pi \mid \mathcal{G})$ ▷ Function describing the cognitive process.
        $\Phi_{u,i} \leftarrow \Phi_{u,i} \cup \phi$
    **end for**
    **return** $\Phi_{u,i}$
**end procedure**

focused on the second explanation step from Miller's framework (Figure 8).

### 3.2.1 Reasons For

To better understand our proposals, we start with the algorithm that generates *reasons for*, i.e., explanations that only *support* the recommendation of interest. In Section 2.4 we described related work on methods that generate explanations from knowledge embeddings predictions [59, 60]. Recalling from Section 2.4, explanations are generated by a search "through" the embedding. The search finds entities satisfying a defined explanation pattern that takes an explanation to be a path containing entities and relations. We use the latter methods to generate *reasons for* recommendations based on knowledge embeddings.

Consider Algorithm 2. In order to generate a *reason for* the input recommendation, we need: an item to be explained $i$, user information $u$, the enlarged weighted knowledge graph $\mathcal{G}$ and the set of paths that specify possible explanations, $\Pi$. Given these input parameters, the algorithm runs through the graph starting at a node $u$ and follows the paths in $\Pi$ with the goal of reaching the recommended item $i$. If it is successful in finding the paths that connect these entities in the specified time limit, the algorithm has found one reason that supports the recommendation. Note that the time limit should be set in the $\gamma$ function, to control how long the search may go on.

### 3.2.2 Reasons Against

We with to propose novel explanation algorithms that are able to present users with reasons that support the recommendations (advantages) and reasons that debunk them (disadvantages). We assume that explanations should assist users in their decision-making processes, so as to let them choose the best possible options in harmony with *their own* opinions, even if their best decision is not to follow recommendations at all. We build on top of the techniques introduced up to now in this section,[1] again emphasizing the need to quickly generate explanations: in a CRS, the user must not wait too much. Our proposals are based on Snedegar's theory of reasons, explored in Section 2.7.

Snedegar presents five of what he calls implementations of reasons *against*, which can an agent contemplating competitive options can use. Those implementations, henceforth called *schemes*, can be summarized as follows:

- **Scheme 1 (S1)**: a reason against an item $A$ is a reason for a competing option;
- **Scheme 2 (S2)**: a reason against an item $A$ is only a reason for NOT $A$ (not for any particular other option);
- **Scheme 3 (S3)**: a reason against an item $A$ is just a reason for the disjunction of the other options (say $B \vee C \vee D$);
- **Scheme 4 (S4)**: a reason against an item $A$ is a reason for each, i.e. all, of the alternatives to it.
- **Scheme 5 (S5)**: a reason against an item $A$ explains (or is part of the explanation as to) why $A$ promotes or respects some objective less well than some other option (this scheme requires one to specify a quantitative objective).

These schemes have been defined by Snedegar at a highly abstract level; we must take them to a concrete form. We present our implementations in the remainder of this section.

Our implementation of S1 generates a reason against a given item by generating reasons for other options. For instance, take the case where the RS has recommended two phones for a potential buyer — phones Red and Green — as in Figure 9. A reason against the Red Phone then would be that the Green Phone has a "Long Duration Battery", while a straightforward reason for it is its "Cutting edge OS".

Scheme S2 is a more delicate one: how to define the negation of an item in the context of recommendations? The vague nature of this question led us to skip this scheme, as we could find no feasible implementation for it in the context of recommendations.

[1]We first proposed these new explanation generation methods, in a joint effort with Gustavo Polleti, using their high-coverage explanation generation [60]. We presented this joint work in the 5th FAccTRec Workshop on Responsible Recommendation at RecSys 2020 [61]. Here we introduce novel practical methods to generate reasons *against* that are included a paper now under review and that is appended to this thesis in Appendix B.

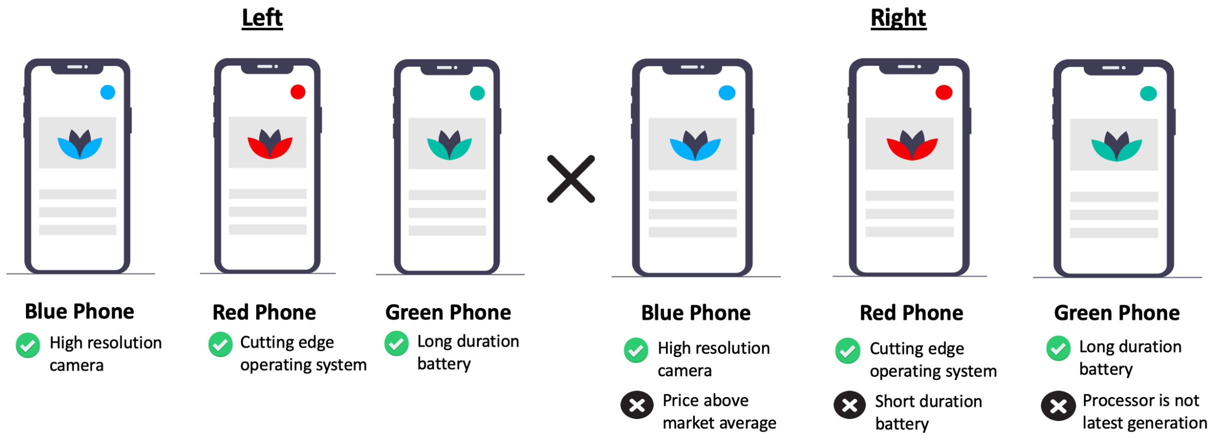



Figure 9: Phone recommendation example. Source: Author [61].

.

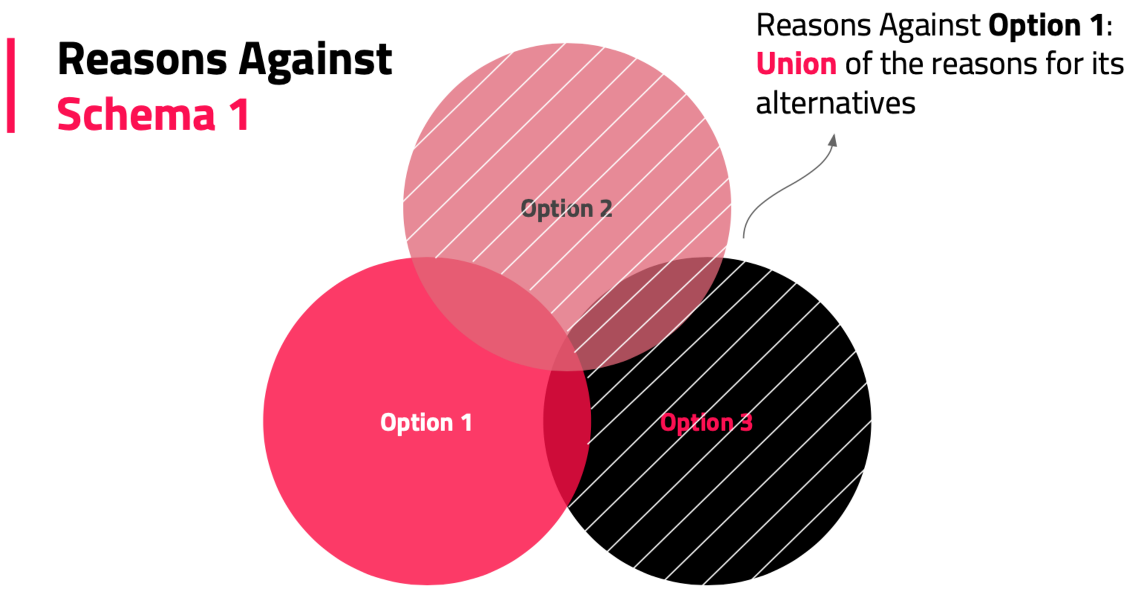



Figure 10: Venn Diagram representing the workings of S1. Source: [61].

Our implementation of S3 goes through competing options, collecting reasons for them that are not reasons for the option of interest; we then trim the list of reasons against to an small number of reasons (the number of reasons must be set). In our running example, we can imagine there is a Blue Phone, and as reasons against the Red Phone, we have that the Green Phone, the Blue Phone or both of them have long duration batteries. In the context of recommender systems, S1 and S3 produce identical reasons against, differentiated by a mere 'or' in the explanation sentence. In Figure 10 we illustrate how Schema 1 works using a Venn Diagram, which also represents S3.

The implementation of S4 resembles that of S3 to the extent that S4 takes reasons for competing options into account (reasons against according to S4 are also reasons against

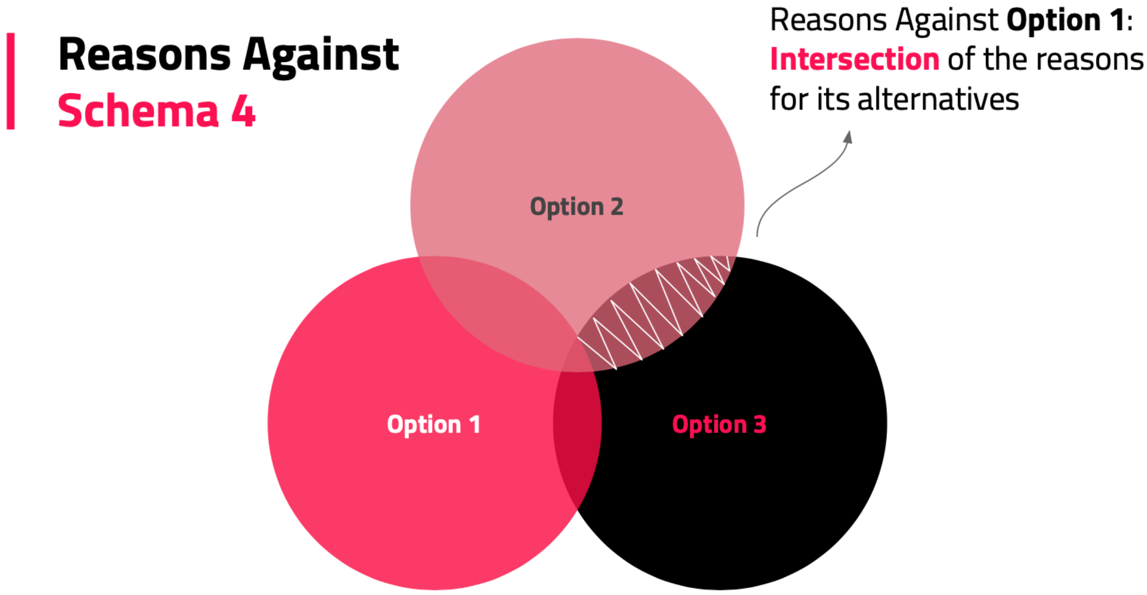



Figure 11: Venn Diagram representing the workings of S4. Source: [61].

according to S3). An example of a reason against the Red Phone using S4 would be that both the Blue Phone and the Green Phone from the example above have adequate battery duration. However, the stringent nature of this scheme, where the intersection of reasons is required, makes it hard to generate reasons against in practical circumstances. We ran tests with our recommender system on topics and classes, and found no explanations whatsoever. That can be explained by the restrictive nature of S4, as seen in the Venn Diagram in Figure 11 by comparing it with Figure10. We can see that, the more options we could have, the more restrictive S4 becomes, since its results come from an intersection operation.

So, suppose a RS recommended $N$ items in an ordered set $\mathcal{I} : \{i_1, i_2, ...i_N\}$ to user $u$. In Schema S1 and S3, we define as reason against an item $i_r$ as the union of reasons for each of its alternatives $\mathcal{I}\backslash\{i_r\}$ that are not reasons for $i_r$ itself. Hence we must iterate over the alternatives, extracting reasons for each one of them $\Phi \leftarrow \Phi \cup \Phi_{u,i}\, \forall i \in \mathcal{I}\backslash\{i_r\}$. We then remove from $\Phi$ the reasons for our recommendation of interest, if any. The remaining reasons $\Omega = \Phi/\Phi_{u,i_r}$ are the reasons against $i_r$ – as presented in Algorithm 3. Regarding the implementation of Scheme 4 (S4), we would follow a very similar procedure, except that instead of considering the union of reasons for its alternatives, we take their intersection. That is, we just replace the line 15 of the Algorithm 3 so as to take the intersection of sets $\Phi \leftarrow \Phi \cap \Phi_{u,i}\, \forall i \in \mathcal{I}\backslash\{i_r\}$.

Scheme S5 has a noteworthy characteristic that differentiates it from the other four. It depends on a quantitative objective that is used as the basis for explanations; this

**Algorithm 3** Explanation Generation using Scheme S1. Source: Author[2].

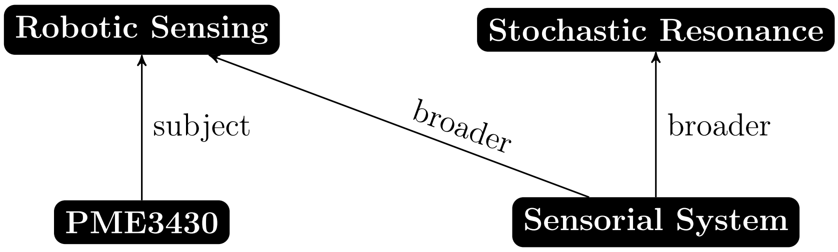
**procedure** REASONS-FOR($i$: item, $u$: user, $\Pi$: set of paths, $\mathcal{G}$: graph)<br>
    $\Phi_{u,i} = \{\}$ ▷ Set of reasons for $i$<br>
    **for all** $\pi \in \Pi$ **do**<br>
        $\phi \leftarrow \gamma(u, i, \pi|\mathcal{G})$ ▷ Function described in Section 2.2.2<br>
        $\Phi_{u,i} \leftarrow \Phi_{u,i} \cup \phi$<br>
    **end for**<br>
    **return** $\Phi_{u,i}$<br>
**end procedure**<br>
**procedure** REASONS-AGAINST-S1($i_r$, $u$, $\mathcal{I}, \Pi, \mathcal{G}$)<br>
    $\Omega_{u,i_r} \leftarrow \{\}$ ▷ Set of reasons against $i_r$<br>
    $\Phi = \{\}$<br>
    $\Phi_{u,i_r} \leftarrow$ REASONS-FOR($i_r$, $u$, $\Pi$, $\mathcal{G}$) ▷ Set of reasons for $i_r$<br>
    **for** $i \in \mathcal{I} \backslash \{i_r\}$ **do** ▷ Iterate over $i_r$ alternatives<br>
        $\Phi_{u,i} \leftarrow$ REASONS-FOR($i$, $u$, $\Pi$, $\mathcal{G}$)<br>
        $\Phi \leftarrow \Phi \cup \Phi_{u,i}$<br>
    **end for**<br>
    $\Omega_{u,i_r} \leftarrow \Phi \backslash \Phi_{u,i_r}$<br>
    **return** $\Omega_{u,i_r}$<br>
**end procedure**

objective is used to determine whether a reason is for or against an option. In a sense, S5 takes that reasons for and reasons against are distinct from each other. We have implemented S5 by assuming that an objective function is known. To illustrate our implementation of S5, consider in our phone example that the user has the objective of extended battery life for her phone. With that piece of information, the RS can present the user with a reason against the Red Phone: "short duration battery".

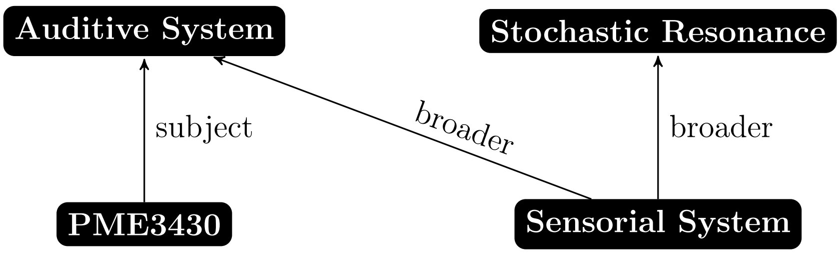
We now turn to discuss the implementation of these schemes in our research domain. Figure 12 depicts a number of explanations generated by our RS, presented in the form of graphs. A student asked our RS for classes about *Stochastic Resonance* and received classes with codes PME3430 and PME3479. The RS found two reasons for PME3430 (Fig. 12a and Fig. 12b) and one for PME3479 (Fig. 12c). Note that both recommendations share the *reason for* depicted in Fig. 12b and 12c (Auditive System), thus it cannot be a reason against for none of them. On the other hand, the reason in Fig. 12a is a reason only forPME3430; therefore, it is a reason against PME3479 and can also be presented as a reason against PME3430.

The implemented Scheme S5 is more complex than the original form proposed by Snedegar [71]. S5 depends on a quantitative objective that serves to measure how well

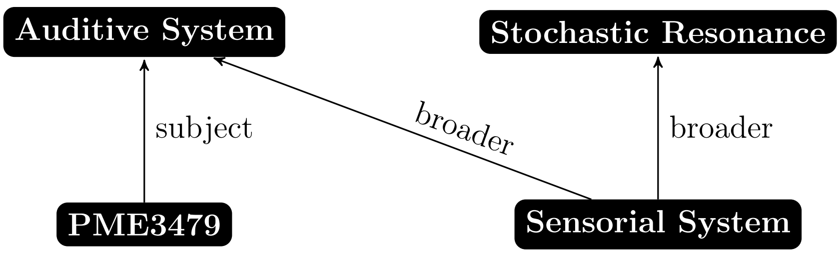
[2]Paper submitted to UMUAI Special Issue on Conversational Recommender Systems:Theory, Models, Evaluations, and Trends.




(a) Reason for PME3430 and against PME3479.




(b) Reason for PME3430.




(c) Reason for PME3479.

Figure 12: Examples of S1 with two reasons for and one reason against the recommendation of PME3430.

the reason does. Actual performance will determine whether the reason is for or against a certain option. For instance, imagine a student who wants to learn about machine learning. Then, both PCS3838 (Artificial Intelligence) and PMR3508 (Machine Learning and Pattern Recognition) are reasonable candidates and will knowingly suffice this student's needs. However, PCS3838 is a broader course which covers many other topics of artificial intelligence than machine learning (ML), while PMR3508 is mostly about ML. In this case, S5 returns that PMR3508 is more about (ML) than PCS3838 and, therefore, `being about machine learning` is a reason that works `for` PMR3508 and `against` PCS3838.

In practice, to determine which of a set of classes is more connected with a topic, we can again rely on the plausibility score calculated by the knowledge graph embedding.

In our effort to implement Scheme 5, we started with a variant with low execution cost. We rely on a search graph with higher-plausibility links placed first. This first implementation is described in Algorithm 4. Using the reasons-for generated previously,

**Algorithm 4** Scheme S5 implemented using response times - proto version

**procedure** REASONS-GENERATION($i$: item, $u$: users, $\Pi$: set of paths, $\mathcal{G}$: graph)
    $\Phi_{u,i} \leftarrow \{\}$ ▷ Set of reasons
    **for all** $\pi \in \Pi$ **do**
        $\phi \leftarrow f(u, i, \pi|\mathcal{G})$ ▷ Embedding link prediction function, Section 2.2.2
        $\Phi_{u,i} \leftarrow \Phi_{u,i} \cup \phi$
    **end for**
    **return** $\Phi_{u,i}$
**end procedure**
**procedure** SCHEME-S5($i_r$, $u$, $\mathcal{I}, \Pi, \mathcal{G}$, $\delta$)
    $\Phi \leftarrow$ REASONS-GENERATION($i_r$, $u$, $\Pi$, $\mathcal{G}$)
    $\Psi \leftarrow$ SORT($\Phi$,by=time) ▷ structure includes the time it took for each explanation
    $\Theta_{u,i_r} \leftarrow \Psi[0]$
    $\Omega_{u,i_r} \leftarrow \Psi[1]$
    **return** $[\Theta_{u,i_r}, \Omega_{u,i_r}]$
**end procedure**

it sorts them by the execution time it took for each reason, in ascending order, then makes the first the reason for the recommendation and the second a reason against the recommendation.

We can refine this algorithm so that the topic of interest itself as the objective. This implementation explains the recommendation by ranking the possible reasons for how much the course promotes learning the topic of interest. Such an implementation of S5 is detailed in Algorithm 5. First, all the reasons for the recommended item being explained $i_r$ are collected, and for each reason, $i_r$ is ranked according to the embedding plausibility score. In the second step, we iterate over all the alternatives to $i_r$ by repeating the same procedure in the first step. If some alternative is better ranked than $i_r$, this alternative is more closely related to that reason than $i_r$, so it is a reason against $i_r$ according to S5. Using the same example as above, in the second step, PMR3508 is better ranked with respect to machine learning; that is why it is given as a reason *against* PCS3838. Noting that REASONS-FOR also has a loop, the algorithm depends on the number of items and on the path length. However, the explanation path length is necessarily short, usually 3.

In short, the generated reasons are sorted in descending order based on their plausibility scores, which are determined with respect to the entity representing the user's quantitative objective, i.e., what they aim to learn.

Please note that, because our explanations are automatically generated and we do not

[3]This algorithm is in a paper submitted to UMUAI Special Issue on Conversational Recommender Systems:Theory, Models, Evaluations, and Trends.

**Algorithm 5** Explanation Generation using Scheme S5. Source: Author[3].

**procedure** RANK($\Phi$: reasons for, $u$: user, $i$: item, $\Theta$: parameters)
  $r = \phi_i \in \Phi$ ▷ The quantitative objective relation
  $\Psi \leftarrow f_r(u, i|\Theta)$ ▷ Assess the quantitative objective as the plausibility score
  **return** $\Psi$ ▷ Measured plausibility score value
**end procedure**
**procedure** SCHEME-S5($i_r$: rec. item, $u$: user, $\mathcal{I}$: rec. set, $\Pi$: paths, $\Theta$: parameters)
  $\Omega_{u,i_r} \leftarrow \{\}$ ▷ Set of reasons against $i_r$
  $\Phi_{u,i_r} \leftarrow$ REASONS-FOR($i_r$, $u$, $\Pi$, $\Theta$) ▷ Set of reasons for $i_r$
  $\Psi_{i_r} \leftarrow$ RANK($\Phi_{u,i_r}, u, \Theta$)
  **for** $i \in \mathcal{I} \backslash \{i_r\}$ **do** ▷ Iterate over alternatives to $i_r$
    $\Phi_{u,i} \leftarrow$ REASONS-FOR($i$, $u$, $\Pi$, $\Theta$)
    $\Psi_i \leftarrow$ RANK($\Phi_{u,i}, u, \Theta$) ▷ Measure of how well $i_r$ meets user's objective
    **if** ($\Psi_{i_r} < \Psi_i$) **then** ▷ Compare if $i$ better meets the objective than $i_r$
      $\Omega_{u,i_r} \leftarrow \Omega_{u,i_r} \cup (\Phi_{u,i} \backslash \Phi_{u,i_r})$
    **end if**
  **end for**
  **return** $\Omega_{u,i_r}$
**end procedure**

further curate them, they might not be always meaningful to the user. But since they are drawn from the knowledge embedding relations, we may suppose that the recommendation itself may not be so insightful in these cases as they might have come from spurious correlations. That remains a topic to be further investigated.

### 3.2.3 Computational Cost

In this section, we analyze the theoretical time complexity of Schemes S1 and S5 with $O$ (Big-oh) notation. This is a mathematical function-analysis notation that allows us to analyze the algorithm's running time in terms of the input size and classify them in different orders of running time growth. A constant time means that regardless of the input size, the running time will be the same. Formally, for a function $g(n)$, $O(g(n))$ is the set of functions [17]:

$$O(g(n)) = f(n) : \exists c, n_0 \in \mathbb{R}, 0 \leq f(n) \leq cg(n) \quad \forall n > n_0 \quad . \tag{3.2}$$

As an example, $5n^2 + 3n - 7$ is in $O(n^2)$ as for $c = 6$ and $n_0 \in \mathbb{R}$, the inequality $6n^2 > 5n^2 + 3n - 7$ holds true for all values of $n$. Table 2 shows the most common orders of growth and how we refer to them.

Let $L$ be the path length, $M$ the number of reasons found and N the number of recommendations given to the user, i.e., the size of $\Phi_{u,i}$. Also, we assume that traditional

Table 2: Common orders of time complexity.

| Time Complexity Order | Name |
|---|---|
| O(1) | Constant |
| O(N) | Linear |
| O(log(N)) | Logarithmic |
| O(N·log(N)) | Linearithmic |
| $O(N^2)$ | Quadratic |
| $O(N^3)$ | Cubic |

knowledge graph embedding models such as TransE, DistMult and ComplEx execute a link prediction in constant time [47]. Therefore, we consider the the plausibility score function $f(\cdot)$ runs in constant time.

Let us now proceed to the analysis, starting with S1. The REASONS-FOR step runs in time $L$, as the for loop executes $L$ times and inside is the function $f$. The appending of reasons runs in constant time for each iteration. The REASONS-AGAINST step runs a for loop $N - 1$ times. Inside the for loop is the previous step, which runs in time $L$. Therefore, this step runs in $L \cdot N$. It is noteworthy that both $L$ and $N$ are relatively small. In our work, we have used $L < 5$ and $N < 10$.

Now focus on the final implementation for S5. Looking at line 10 in Algorithm 5, we see that the first step uses REASONS-FOR, similarly to S1. Then S5 has a for loop that runs also $N - 1$ times: order $N$. Inside the loop, we have REASONS-FOR ($L$) and RANK. RANK, on the other hand, runs a sorting algorithm that we may assume to be Heap or Merge Sort. Hence, it would run in time complexity $O(M \cdot log(M))$.

The for loop in SCHEME-5 runs, therefore, in time $O(N \cdot (L + M \cdot log(M)))$ and the complete algorithm, in time $O(L + N \cdot (L + M \cdot log(M)))$.

We thus conclude that our proposals are feasible from a time complexity standpoint. In Chapter 5 we present experiments that confirm this claim.

# 4 SOFTWARE DESIGN AND IMPLEMENTATION

*"Inventions are, above all, the result of stubborn work, in which there must be no room for discouragement."*

-- Alberto Santos Dumont

In order to evaluate our proposals, we built a real explanation-capable conversational recommender system we call Calisto [1]. We describe our implementation in this chapter. First, we present the requirements, then the specification and finally the development of the system.

## 4.1 Requirements

In this section, we present the functional and non-functional requirements of the system. Requirements are physical or operational needs that the system must satisfy to achieve its end goal [62]. Functional requirements map into a specific function or functionality in the system and may involve calculations, data manipulation and processing in the actual implementation. In contrast, a non-functional requirement define a particular property of the system behavior [7].

We introduce the functional requirements of the system in Table 3 and explore them in the next paragraphs.

The system's primary function is to provide recommendations to students, who are the end-users. In our case, the system gets the field of study about which the user wants to learn. The system must identify the relevant classes for this field of interest and must make suggestions to the student. The system must then find relations between the suggestions to the field of interest, i.e., it must find some connection between them. After that, the system must present these relations to the student.

At this point, it is crucial to specify how the system will communicate with the student, i.e., the nature of the human-computer interface. As we are building a conversational

[1]Calisto is one of Jupiter's moons and we chose this name because the undergraduate courses management system at USP is called Jupiter. Also, a previous version of Calisto, with only explanation for, was called Ganymede - another of Jupiter's moons.

Table 3: Functionality table for the recommendation and explanation module.

| **Objective** | ***Requirement*** |
|---|---|
| Suggest units of study | Identify student's field of interest.<br>Identify courses belonging to the student's field of interest.<br>Suggest a course according to field of interest. |
| Explain suggestions given | Find relations from suggestions to field of interest.<br>Explain relations to student. |
| Communicate with student | Availability of a public user interface for user interaction.<br>Understand human language in Brazilian Portuguese.<br>Generate human-readable sentences for recommendations and explanations. |

agent, users should communicate in verbal or written sentences, and the system likewise. As it is possible to easily convert text to speech using a tool specifically for that (for example, Amazon Text-to-Speech or IBM Watson), the challenge is reduced to generating human-readable sentences.

Our target users are students from the University of São Paulo, a Brazilian university where most students speak Portuguese, so the language must be Brazilian Portuguese. The availability of a public interface for user interaction is a key requirement.

A conversational agent, being interactive, must have a quick response time. The Dialogflow tool, used for interfacing the client with the recommender system, has a 5-second *timeout* specification by construction. That is, if the *back-end* does not respond within that time-frame, the Dialogflow interface presents the user with the message that it could not understand the question or that it cannot explain that recommendation. Due to the nature of human information processing, i.e. the limits of the short term memory, we must keep the time interval between interactions short [13, 52] and that, as pointed out above, is built into the conversational platform we use. Hence we require that response time for recommendations and explanations must be below 5 seconds.

Other essential requirements of the system are related to the quality of the explanations. The system has the following non-functional requirements:

1. Explanation coverage above 70%;

2. Average explanation support equal or over 1.0.

These requirements focus on the quality of the explanations generated by the system, measured by coverage and support. We define *coverage* as the fraction of recommendations

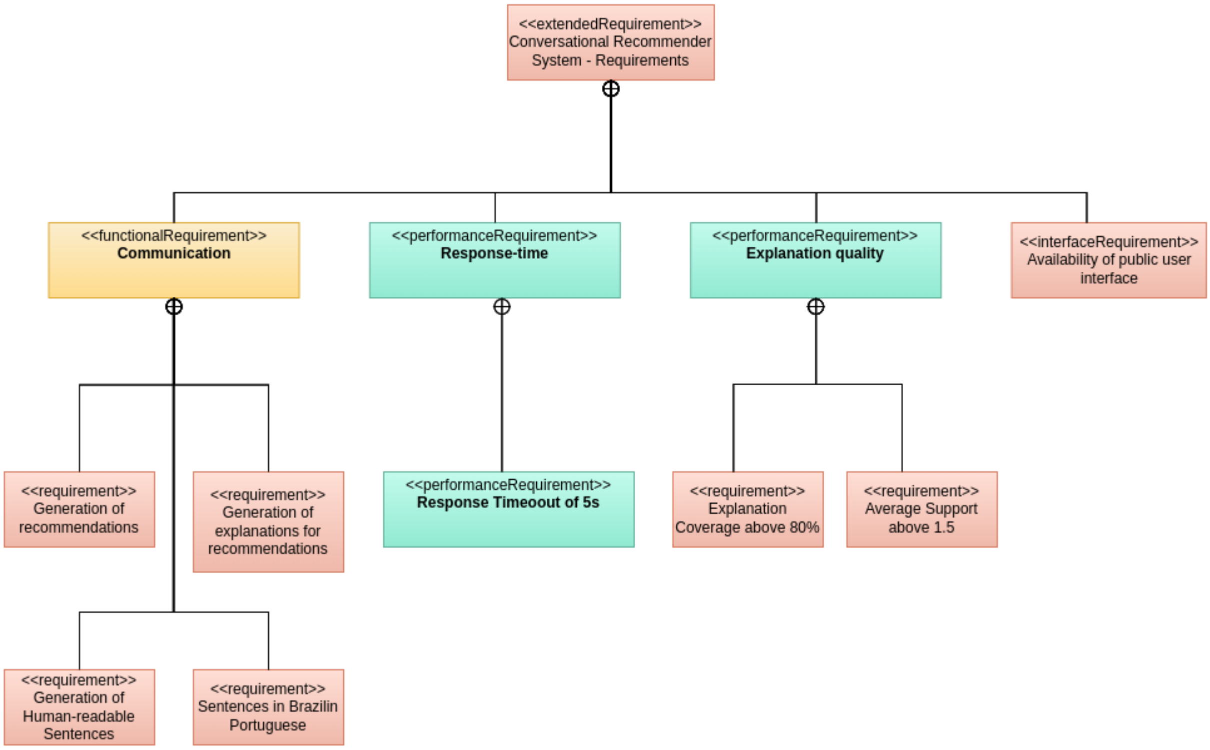



Figure 13: SysML Requirements diagram for Calisto. Source: Author.

for which we can find at least one explanation and *average support* as the number of reasons we can find to explain (may it be in favor or against) a given recommendation [60, 83]. The values specified about are sensible as they are achievable and sufficient, according to previous works [58, 61].

To summarize the requirement explanations outlined in this chapter, see the requirements Diagram in Figure 13, which illustrates the requirement specification for the system following SysML [23, 72].

## 4.2 Software Design

In this section, we present the architecture design for the recommender system, as well as other diagrams and context relevant to understanding its workings and implementation. The proposed architecture, seen as a high-level architecture diagram in Figure 14, has three main components so-called the Front-end, the Back-end and the Database. There is also an actor, the student, who interacts with the system through its interface.

The interface is the conversational agent, enclosing Natural Language Understanding (NLU) software that can process the user input into a defined intent (see next section). With additional relevant information, this intent is passed then onto the Back-end.

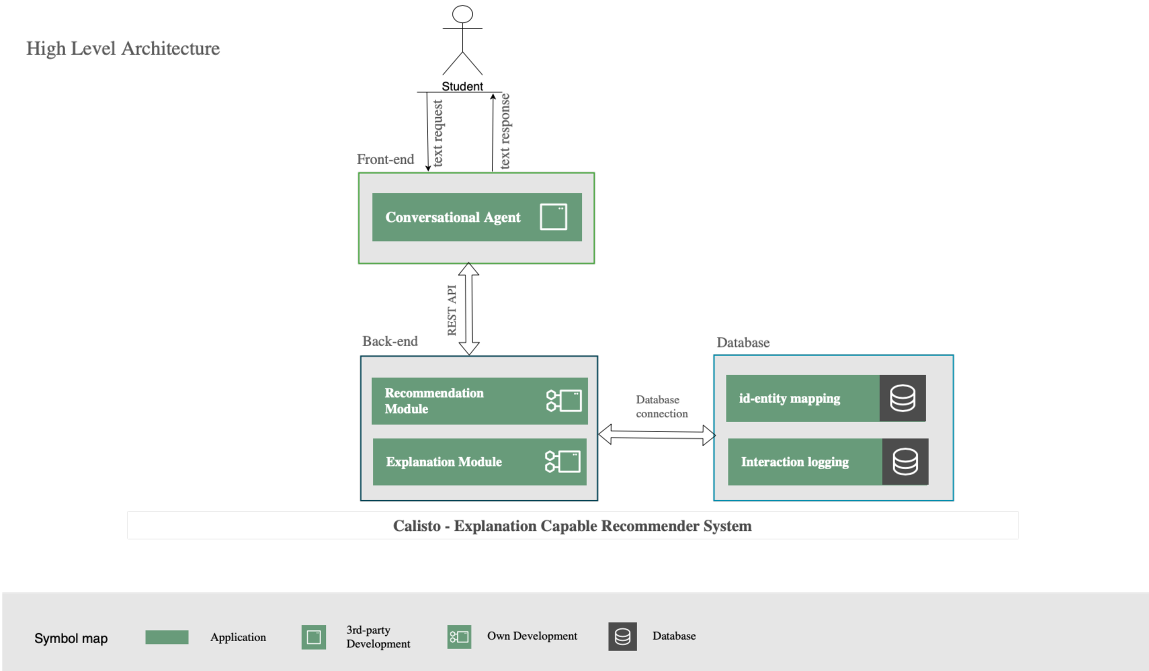



Figure 14: High level architecture diagram for Calisto. Source: Author.

The Back-end is composed of the recommendation and explanation modules. They are the system's backbone, providing the critical functionalities of recommendation and explanation over the recommendations given. These modules are the core of this work.

The third component, the Database, serves as a helping hand to the Back-end for two purposes. The first is mapping a particular entity in the knowledge graph from its numeric id to a human-friendly name used for both recommendations and explanations in the textual response to the student. The second purpose is to log the interactions for debugging, testing and evaluation of interaction results both in development and research surveys with real users. We have used a real-time cloud database. We calculate and evaluate the research results from the stored data that also works as the repository of data for research carried out using this system.

From the logical standpoint, we may visualise the system by its data flow, viewed through the Sequence Diagram in Figure 15. This diagram shows how the conversational agent mediates the communication between the student (actor) and the recommender and explainer modules. Requests are sent to the corresponding Back-end module through the agent, which processes and sends the relevant response. Again, the agent deals with interfacing this information with the student.

As noted previously, we built our chatbot interface using Dialogflow (available at dialogflow.cloud.google.com), a freely distributed package that supports the Portuguese language. In that platform, we created the chatbot with the intents presented in Table 4,

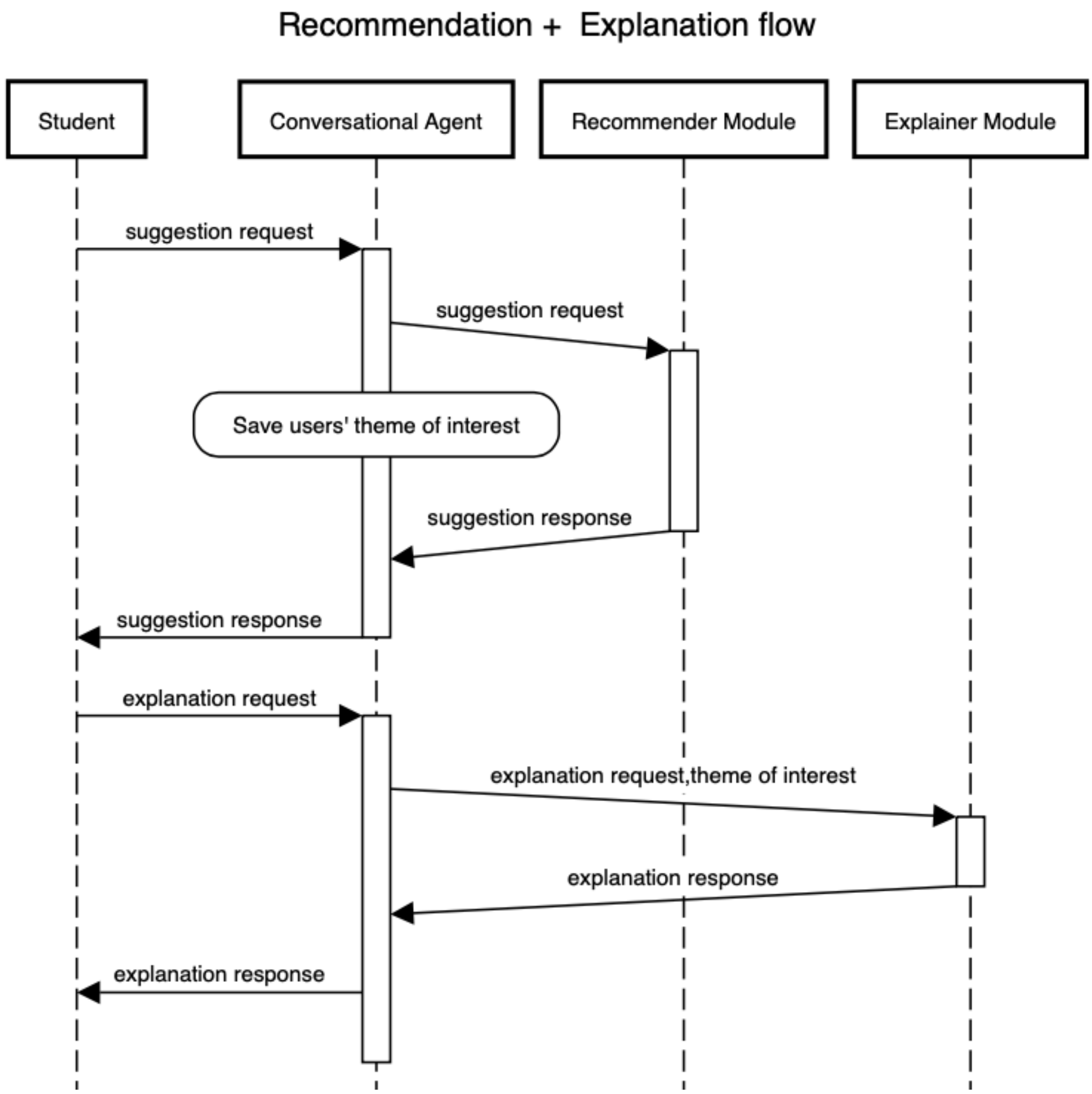



Figure 15: Sequence diagram for interactions between users and Calisto for recommendation and explanation. Source: Author.

based on previous works that implemented chatbots for recommending units of study [19, 57]. These intents map to a function inside the system, which either provides suggestions/explanations helping the user better deal with the system or provides feedback for the interaction.

The recommendation (or recommender) module is tasked with receiving a suggestion request, a theme and a path and then finding, parsing and sending the recommendations back to the requesting agent.

State-of-the-art recommendation techniques learn affinity between users and items from previous data using latent variable models, which often depend on matrix factorization and embedding techniques [28, 30, 33]. These approaches work by transforming item

Table 4: Intents implemented in the conversational agent for Calisto.

| **Intent** | **Meaning** |
|---|---|
| GetCourseSuggestion | Requests a recommendation |
| GetCourseSuggestion - explain | Explains a recommendation |
| Help | Guides user to the system functionalities |
| Feedback - transparency | Acquires user feedback (transparency metric) |
| Feedback - persuasion | Acquires user feedback (persuasion metric) |
| Feedback - engagement | Acquires user feedback (engagement metric) |
| Feedback - trust | Acquires user feedback (trust metric) |
| Feedback - effectiveness | Acquires user feedback (effectiveness metric) |
| Feedback - comment | Acquires user feedback (comment metric) |
| Thanks | Displays farewell message |

attributes into a latent space so that related objects have their distances minimized as a result [51].

Explanations are saved in a dictionary, and must then be translated to natural language so that the user may understand them. For that purpose, a sentence template has been adopted, which can be represented in the Backus-Naur form as seen below:

<explanation>::=<sentence>|<sentence><compl_sentence>
<sentence>::=‘Recomendo ’<entity3>‘ sobre ’<theme>‘, pois ’<entity3>‘ é sobre ’<entity2>‘, que é da categoria ’<entity1>‘ assim como ’<entity0>
<compl_sentence>::=‘No entanto, ’ <neg_sentence|less_sentence>
<neg_sentence>::=‘ela não é sobre ’<themes>‘ como ’<reasons_against><verb>
<less_sentence>:==‘é menos relacionada a <themes>‘ do que ’ <subject_against>. <verb>::=‘é|são’
<themes>::=<theme>|<themes>‘ ou ’<theme>

The `theme` above is the theme provided by the student in the chatbot when asking for a recommendation. The variable `entity0, entity1, entity2` and `entity3` are the entities in the explanation path found as explained in the previous section, i.e., they are the forming parts of the reason for. Likewise, `reasons_against` are the set of reasons against the recommended item, found according to Schemes 1 or 5 and may well be a singular reason.

The sentence `compl_sentence` is only present in schemes that produce also *reasons against*. Additionally, `neg_sentence` is applied in S1 while `less_sentence` is in S5.

## 4.3 Implementation

This section describes some of the steps we took to implement a recommender with explanation capability using the algorithms we proposed.

We relied on a large knowledge base known as USPedia. The data used to feed the system were gathered and organized into USPedia in a prior effort by our research group [60]. More specifically, the recommender uses the links from professors to courses they teach and to articles (papers) they published to help in inferring categories and topics for courses. Note that professors may not necessarily teach about all topics that they have published about, but we suppose there is a correlation. At this point, it is convenient to briefly describe USPedia. We reproduce part of its description below, from the article which firstly described it [60, p. 5]:

> The KG (USPedia) structure consists of five types of entities: learning object, professor, concept, and category. The learning object includes both graduation courses offered by the university and articles authored by faculty members. Concepts and categories represent an ontology for the learning-objects content. Thus, we say that a faculty member is involved in multiple learning objects in which they can either teach a course or author an article and that each learning object is about multiple concepts.
>
> To build the KG, we opted for using an automated semi-structured approach (Nickel et al. 2015) as it has also been employed by many popular large-scale KGs, such as DBpedia (Auer et al. 2007) and YAGO2 (Hoffart et al. 2013). The selected approach automatically extracts information from semi-structured data, like info-boxes, via rules or regular expressions. Firstly, we collected the description and 543 teachers' names for 1740 engineering courses from the university graduation support website.
>
> We also retrieved from the academic repository Scopus 7648 articles authored by the faculty members. Finally, we performed entity linking to DBpedia using the articles' keywords and course descriptions content, so the knowledge base incorporated DBpedia's hierarchy of concepts and categories.
>
> The resulting Knowledge Graph has 34182 entities, 3 relations and 152468 triples.

The training of the knowledge graph embedding was carried out according to the model TransE, as seen previously in Section 2.2.2. To train TransE we chose OpenKE.

Table 5: Training parameters for the knowledge graph embedding. Source: Author.

| **Parameter** | **Value** |
|---|---|
| model | TransE |
| work_threads | 8 |
| nbatches | 100 |
| margin | 1.0 |
| bern | 0 |
| train_times | 1000 |
| dimension | 500 |
| ent_neg_rate | 1 |
| rel_neg_rate | 0 |
| opt_method | SGD |
| $\alpha$ | 0.001 |

The training parameters were tuned for better performance according to a grid search procedure as described in [50] and were finally set according to the values in Table 5.

We designed and implemented a REST API to allow the chatbot to communicate with the back-end services - recommendations and explanations. For that, we used a Python module for web development, Flask. In the production environment, we replaced the Flask server with the production-grade server Waitress, as suggested by Flask's own documentation [2]. Additionally, for logging, we used a tool called Logger to format and output parameters in some parts of the interaction.

The resulting system is described in Appendix A, with some screenshots of the front-end and the back-end running.

[2]https://flask.palletsprojects.com/en/1.1.x/deploying/

# 5 EXPERIMENTS AND RESULTS

*"In preparing for battle I have always found that plans are useless, but planning is indispensable."*

-- Dwight D. Eisenhower

The most desirable evaluation of a recommender system is a test with real users. Aside from that, typical machine learning evaluation techniques can be coupled to enhance confidence in the system's robustness and efficiency [65]. In this work, we coupled tests with real users and several offline objective evaluations of the proposal, which together give a fuller description of the overall system performance.

This chapter reports the experiments we realized and discusses the results we obtained through those experiments. We first present an analysis of the algorithms time complexity in practice, focusing on their run time to generate explanations. Then, we present practical off-line evaluations regarding their output coverage and support. Finally, we report on the experiments performed with human subjects.

We summarize this evaluation method in Figure 16, based on Doshi-Velez and Kim's Functional-grounded, Human-grounded and Application-grounded evaluation framework [20]. Note that, although the specificity and cost increase in the order presented, we timed the evaluations in an order of Functional, Application and Human-grounded.

## 5.1 Practical Algorithm Analysis

In Section 3.2.3, we analyzed the algorithms time complexity using Big-O notation. In this section, we focus our analysis on algorithms for S1 and S5, including a preliminary approach to implement S5 which we call S5(v0). We describe an experiment that evaluates their execution time, also taking into account their coverage and support.

We set up an experiment with a sample size of 100 themes, chosen according to a uniform distribution over the total of 6814 entities available in the training set for the knowledge base. We proceeded to request recommendations for each of them and then generated explanations for each of those recommendations using algorithms S1, S5 and

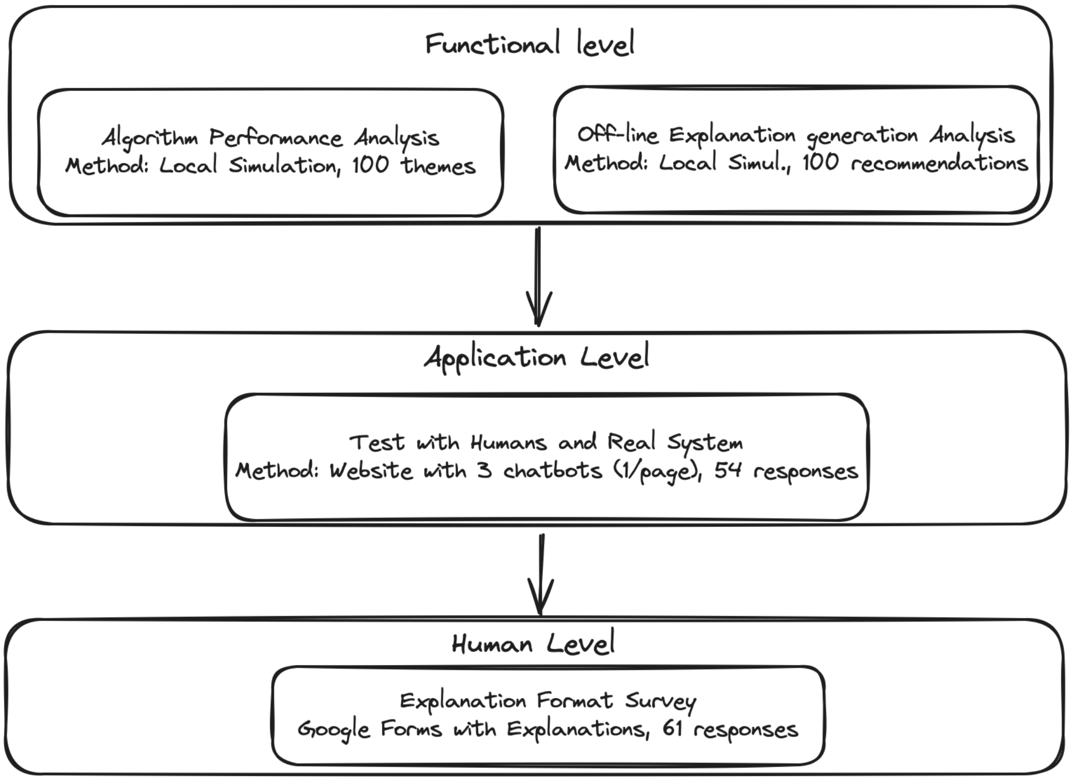


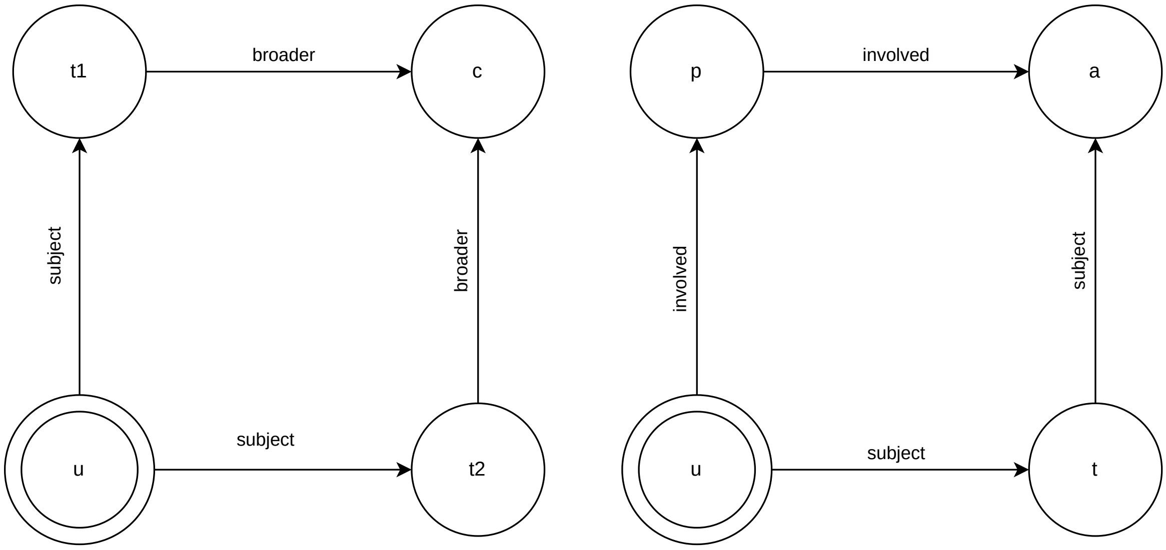

Figure 16: Evaluation method illustrating all the experiments we ran.

S5(v0). For each algorithm, we tried the two different explanation paths seen in Figure 17a and Figure 17b, henceforth called P1 and P2.

For each algorithm, we evaluated results for thirty recommended topics, each receiving five recommended courses. P1's rationale is that if a specific course (u) covers a topic (t1), whose category (c) falls under the same as some other topic (t2), then u likely covers the topic of interest t2 for which we want an explanation. For instance, say we would like to learn about pandemics, and we are recommended BIO3100 - Tropical Diseases and Epidemics, which covers epidemics, under the category epidemiology. Then, we offer the following explanation: we recommend BIO3100 about pandemics because it is about epidemics, which is under the category epidemiology as pandemics is.

P2, on the other hand, conveys the notion that if u covers the topics of an article (a), which was published by a professor (p) who also teaches u, then u should also be about topic t. For example, if a co-author of "A new Covid-19 crisis: Domestic abuse rises worldwide", which is about domestic abuse, lectures a course SOC0101 - Social Aspects of Violence, this path assumes this unit covers the topic of domestic abuse.

In an offline experiment, we found that the results vary slightly for paths P1 and P2. In Table 6, we show coverage, support and execution time for the algorithms we




(a) Topic explanation path (P1). (b) Professor explanation path (P2).

Figure 17: Explanation paths P1 and P2 used in this experiment.

Table 6: Coverage, support and execution time for each algorithm using explanation path types P1 and P2. Coverage is given in percentage of classes for which an explanation was found, support is the average number of explanation found for those units and time represents the mean execution time per explanation in seconds.

| **TransE Embedding Results** | | | | |
|---|---|---|---|---|
| **Algorithm** | **Path** | **Coverage** | **Support** | **Time (s)** |
| **Reasons For** | P1 | 71.83% | 24.6 | 3.920 |
| | P2 | 71.83% | 9.7 | 3.951 |
| **S1**[1] | P1 | 71.83% | 1.0 | 3.931 |
| | P2 | 71.83% | 1.0 | 3.953 |
| **S5(v0)**[1] | P1 | 20.42% | 1.0 | 3.491 |
| | P2 | 21.13% | 1.0 | 3.944 |
| **S5**[1] | P1 | 65.49% | 1.0 | 3.931 |
| | P2 | 65.49% | 1.0 | 3.959 |

presented, considering both path types. In terms of coverage, we only note a difference in the prototype version of S5, while there is a difference in support in the case of the Reasons For algorithm. In the other cases, all algorithms were designed to have support equal 1.0. Finally, in terms of execution time, we find a considerable difference again only in the case of S5(v0). That can be explained by the very characteristic of this algorithm, that uses the execution time as a proxy to the plausibility of the explanations found. This means that, according to this method, explanations found through P1 are more plausible, even though we are able to cover slightly less recommendations.

[1]These algorithms are designed to yield at most one explanation for and one explanation against, so support is either 0 or 1.

## 5.2 Offline Evaluations

The implementations were also evaluated from two other perspectives: (1) the fraction of recommendations for which we can find at least one explanation (referred to as *coverage*) and (2) the average number of reasons we can find to support/attack a given recommendation (referred to as *support*) [60, 83]. These metrics offer a glimpse at the workings of Calisto in a real-world scenario from an objective perspective. The simulated interactions were built by asking for the Top-4 recommendations of randomly sampled 100 cases. Next, we used our implemented explanation schemes to retrieve both reasons for and against each recommendation previously found.

Regarding *reasons for*, Table 7 shows that we obtained 79.33% coverage and a support mean of 2.0, similar results to those reported in previous works [60, 83]. As for *reasons against*, we ran our experiments considering Schemes S1, S4 and S5. Both the coverage (85.1%) and support (2.3) obtained for S1 are higher than those from *reasons for.* This result was expected since the S1 implementation considers more aggregated reasons for alternatives than it removes from the recommendation being explained.

On the other hand, Scheme S4 could *not* generate a single reason against at all (coverage 0%!) in our settings. This happened because Scheme S4 requires that a reason against an option must be a reason *for all* of its alternatives, imposing a restriction so rigorous that it is, in fact, unfeasible in practice. This restrictive nature of S4, due to the intersection operation, is further illustrated in Figure 11, which can be compared with Figure 10. In S1 the union of *reasons for* alternatives is taken to be *reasons against*, while in S4 *reasons against* are formed by the intersection of the *reasons for* the alternatives.

The support and coverage results for S5 are also shown in Table 7. The average support of 1.0 is expected since, by construction, Scheme 5 yields a single reason against and a single reason for each recommendation. As for coverage, we obtained the value of 83%, which is a similar score to S1's. When leaving out the explanations which contained only reasons for, S5's coverage reaches the value of 61.7%. We hypothesize that this is due to embedding incompleteness, which led to 17% of missing reasons against, i.e., links between subjects and courses. This difference remains to be further investigated.

[2]Partly published in [61]. Coverage results for S5 would be 61.7% if explanations containing only reasons were left out.

Table 7: Coverage and Support results for Schemes S1 and S5. Source: Author[2].

| Explanation Type | Coverage | Support |
|---|---|---|
| **Reason For** | 79.3% | $2.0 \pm 1.0$ |
| **Reason Against (S1)** | 85.1% | $2.3 \pm 1.4$ |
| **Reason Against (S4)** | 0% | - |
| **Reason Against (S5)** | 83.0% | $1.0 \pm 0$ |

## 5.3 User Test

In a user test, we gather a sample of users and request them to perform some actions and evaluate the system responses according to some pre-established metrics. We decided to evaluate our proposal in five dimensions - transparency, persuasion, engagement, trust, and effectiveness. Besides that, we explicitly asked the users to comment on their overall impression of the system.

Transparency measures how much the explanation clarifies the process of finding the recommendation. Persuasion measures how likely the users are to accept the suggestions they receive. Engagement indicates how the explanation helped them learn more about the recommended items. Trust, in its turn, is a figure that represents the confidence the users have in the correctness of the recommendations. Finally, effectiveness is an overall gauge of the recommender system as a whole [73].

We chose to represent all those metrics in a Likert scale, which ranges from one to five in order to avoid central tendency [40] from 1 to 5 (standing for "Strongly disagree", 2 "Disagree", 3 "Neither agree nor disagree", 4 "Agree", and 5 "Strongly agree").

Our experiment took 54 subjects, all of which were final year undergraduate engineering students, and asked them to evaluate the three RSs implementations, one displaying only reasons for recommendations, the other displaying reasons against for and against according to S1 and the other displaying reasons for and against according to S5. Additionally, a toy recommender system allowed users to get familiar with the interface since that was not part of the evaluation. The webpage set up for the experiment may be visualised in Figure 25 and Figure 26, where one sees the experiment flow at the top of the page and the actual chatbot at the bottom, with some clarifying text in the middle.

Each subject went through a series of tests for each of the explanation implementations, each type in a different recommender, served by different links to the human subjects. For each of those systems, we asked questions to evaluate the implementation's score for each metric we defined above. At the end of the interaction, each subject

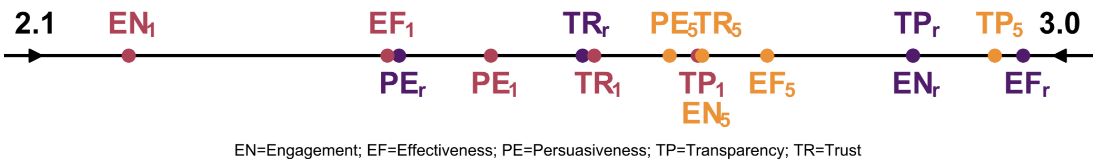



Figure 18: Visual representation of the average scores for the explanation metrics comparing Reasons For, S1 and S5. Source: Paper under review; please refer to Appendix B for the paper.

could write a short, free-format text with remarks about the systems, with critiques or suggestions as they desired.

The results we obtained can be viewed in Figure 18, which depicts the explanation metrics scores on a continuum representing the Likert scale visually. The same results are represented in Figure 19 as a bar chart for a more intuitive visualization. We can observe that the conversational recommender system employing Schema S5 was the best approach from the user's perspective by means of transparency ($\mu_{transparency} = 2.94$), trust ($\mu_{trust} = 2.68$) and, particularly, persuasion ($\mu_{persuasion} = 2.65$). Furthermore, Schema S5 was better than S1 with respect to all measures. Other effects of introducing reasons against in explanations are the drop in engagement and effectiveness, i.e., the baseline (PRED/reasons for) reached the highest ratings in both metrics ($\mu_{effectiveness} = 2.96$ and $\mu_{engagement} = 2.87$)[3].

## 5.4 User Survey - Explanation Format

By taking into account the importance of user understanding of our explanations, we also run an experiment to get insights into the most suitable format for explanations. The experiment followed the scoping, planning, operation, analysis, and evaluation phases proposed in [78]. The experiment aimed to analyze explanation formats for the algorithm S5 to decide which explains better concerning effectiveness from the students' point of view in the context of a subject-object study with undergraduate students from the Escola Politécnica at the Universidade de São Paulo.

The context of the experiment was a machine learning course offered at the Universidade de São Paulo. The experiment was run offline, with students using Google Forms for the questionnaire, and presented subjects with a real and a toy problem. The setting was class recommendation for an engineering student, while the toy problem dealt with recommending classes for someone wishing to learn about investments. For this experi-

[3]Results to be published in paper under review. Please refer to Appendix B for the paper.

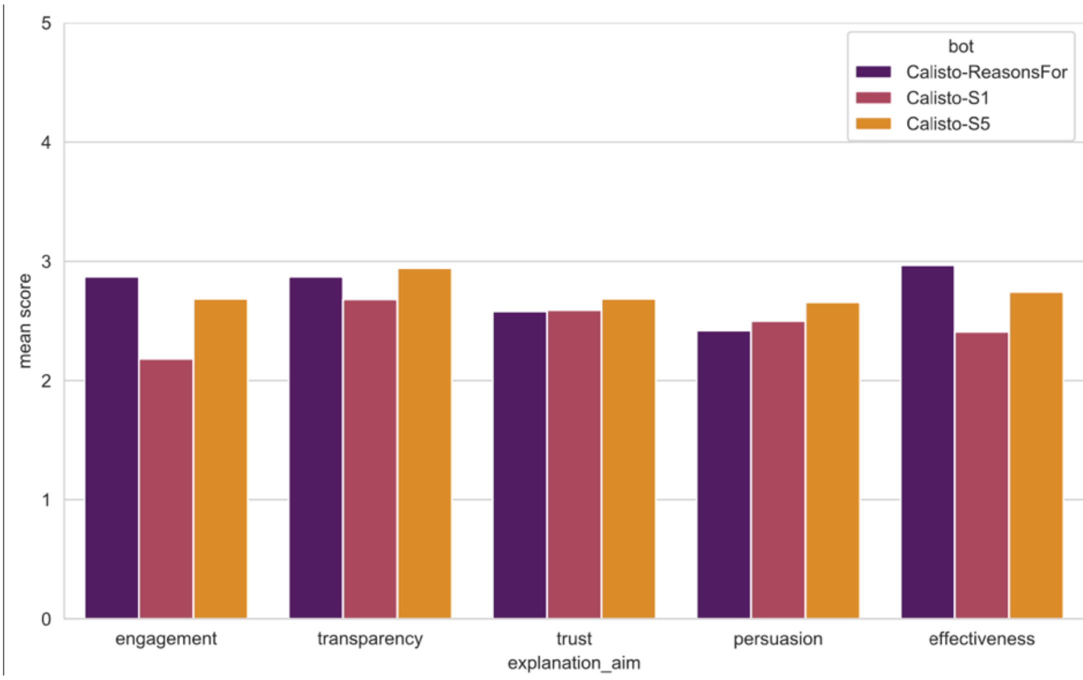



Figure 19: Bar chart with the average scores for the explanation metrics. Reasons For, S1 and S5 results are in purple, red and yellow, respectively. Source: Paper under review; please refer to Appendix B for the paper.

ment, we surveyed 62 students and obtained 61 valid responses. There was only one that responded too quickly in comparison to all others and gave meaningless responses in the open questions.

Our hypothesis with the experiment was that an explanation with more information would be more effective. To verify that, we introduced three different explanation formats, which we nick-named Gamma ($\Gamma$), Kappa ($K$) and Xi ($\Xi$). The order in which these three different formats appeared was randomly assigned to each participant, and we chose the naming so as not to provide information about order or importance. Note: explanation format $\Xi$ is the one we used for the experiment in Section 5.3 and that we described in Backus-Naur form in Section 4.2.

$\Gamma$: I recommend COURSE1 as the most adherent to TOPIC1. However, it covers the subject SUBJECT less than COURSE2, which is also about TOPIC1.

$K$: I recommend COURSE1 about TOPIC1, since it is about TOPIC2 from the category CATEGORY as TOPIC1. However, COURSE1 covers the subject SUBJECT less than COURSE2.

$\Xi$: I recommend the course COURSE1 as the most adherent to TOPIC1, since COURSE1 is about TOPIC2 from the category CATEGORY as TOPIC1. However, it covers the subject SUBJECT less than COURSE2, which is also about TOPIC1.

As one can see above, there is a decreasing sentence size, a proxy for quantity of information, from $\Gamma$ to $K$ and $\Xi$. Note that they were always randomly assigned in the experiment.

In our setup, the independent variable is the explanation format F, where F $\in \{\Gamma, K, \Xi\}$. The dependent variable is the Effectiveness $e$. So, we want to compare the effectiveness means $\mu_e$ for each value of the explanation format. For that, considering we expect larger sentences to yield a higher effectiveness, we have three unilateral hypothesis tests to evaluate - T1, T2 and T3 below.

$$T1 = \begin{cases} H_0: & \mu_{e\Xi} = \mu_{e\Gamma} \\ H_1: & \mu_{e\Xi} > \mu_{e\Gamma} \end{cases} \tag{5.1}$$

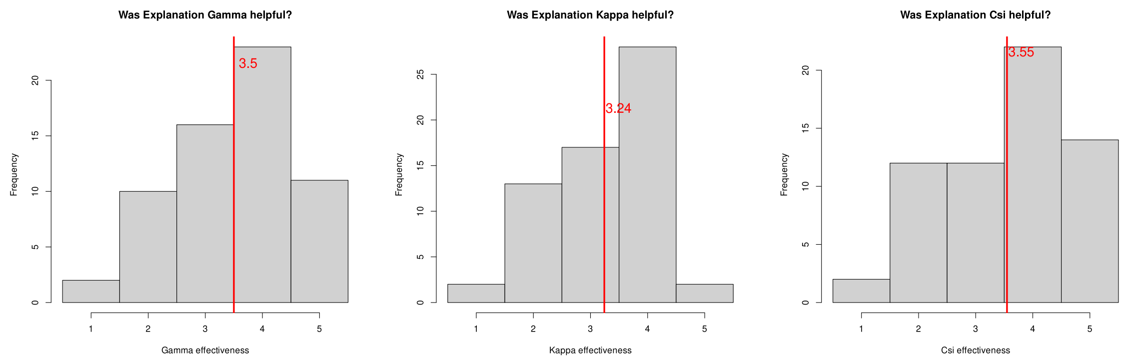
$$T2 = \begin{cases} H_0: & \mu_{eK} = \mu_{e\Gamma} \\ H_1: & \mu_{eK} > \mu_{e\Gamma} \end{cases} \tag{5.2}$$

$$T3 = \begin{cases} H_0: & \mu_{e\Xi} = \mu_{eK} \\ H_1: & \mu_{e\Xi} > \mu_{eK} \end{cases} \tag{5.3}$$

With these three hypotheses to test, we created a questionnaire to carry out our experiment. The complete questionnaire we applied is available in Appendix B.2. To apply it to the students, we followed a simple random sampling, in which the order of the questions was shuffled by Google Forms. We analyze the results we gathered in the remainder of this section.

First, let us consider the data in the three histograms in Figure 20. Each histogram presents results from a question as to whether an explanation was helpful, following a Likert scale. In red, we show the average response that represents the mean effectiveness $\overline{e}$. In Figure 20c we have $\overline{e}_\Xi = 3.55$, in Figure 20b, $\overline{e}_K = 3.50$, and in Figure 20a, $\overline{e}_\Gamma = 3.24$.

Taking a 5% significance level and noting that we have data from paired experiments, we proceed to test our hypotheses. We carried a more conservative, paired two-sample T-test (Welch two-sample T-test) with different variances for each group and also ran a Mann-Whitney-Wilcoxon Test which does not assume normality in the data. As we show in Table 8, tests T1 (Equation 5.1) and T2 (Equation 5.2) led to no rejection of $H0$ for both statistical methods. On the other hand, T3 (Equation 5.3) led to rejection of $H_0 : \mu_{e\Xi} = \mu_{eK}$ in favor of $H_a : \mu_{e\Xi} > \mu_{eK}$.




(a) Explanation $\Gamma$. (b) Explanation $K$. (c) Explanation $\Xi$.

Figure 20: Was the explanation helpful? Source: Author

Table 8: Statistical test results for tests T2, T3 and T1. Source: Author.

| **Statistical Method** | **Test** | **p-value** | **Reject $H_0$** |
|---|---|---|---|
| Paired Welch two-sample T-test | T1 | 0.404 | No |
| | T2 | 0.925 | No |
| | T3 | 0.039 | Yes |
| Paired Mann-Whitney-Wilcoxon Test | T1 | 0.379 | No |
| | T2 | 0.902 | No |
| | T3 | 0.034 | Yes |

Looking at the sample distributions in Figure 20, we see that there is a high concentration of responses around the mean value of 3, which in a Likert scale indicates a level of uncertainty. In cases of Figure 20a and Figure 20c, we also note that the histograms look similar besides their means being approximate, i.e., $\overline{e}_{\Xi} = 3.55$ and $\overline{e}_K = 3.50$. So, in these two cases we see that, coherently with the statistical test results in Table 8, it is not possible to differentiate between the two explanation formats $\Gamma$ and $\Xi$ according to their effectiveness.

In the case of explanation $K$, we have $\overline{e}_{\Gamma} = 3.24$ and see that there are less responses with a score 5, which show that there seems to be a considerable difference between the effectiveness of this explanation format and the other ones, specially format $\Xi$. This difference is statistically confirmed for the comparison of $\Xi$ and $K$, as we can see in Table 8. In both statistical tests, the more conservative T-test and the non-parametric Mann-Whitney-Wilcoxon test, we find we may reject the null hypothesis in favor of $\Xi$ yielding a higher effectiveness than $K$.

For higher clarity, however, we asked students which explanation they found to be most helpful. We see the result in Figure 21, showing that the winning format was $\Xi$ - the most complete explanation. Noting that the order in which explanations appeared

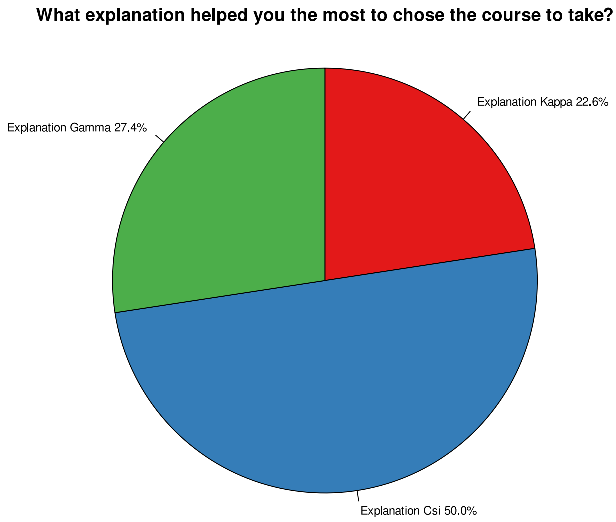



Figure 21: What was the most helpful explanation? Source: Author.

was random, we conclude that, indeed, explanation format $\Xi$, the one that provides the most information, is favored by the students.

We also note that responses for the most helpful explanation were aligned with the Likert-scale responses for effectiveness. The order of preference $\overline{e}_{\Xi} > \overline{e}_{\Gamma} > \overline{e}_{K}$ is maintained in answers across the questionnaire. This order requires some further analysis, which we do next.

We may describe these three explanation formats according to how complex or complete they are, which we can represent by their length and entropy. The length allows us to compare the number of words in the sentence, while the entropy measure the quantity of information in each sentence.

First, note that this we analyzed the original versions of the explanation formats, in Brazilian Portuguese, and not the translation to English as written above. To examine this analysis in detail, please refer to Appendix B.2. To calculate the length, as our purpose is to measure how much more effort the student would need to employ in order to read the sentence, we use the number of words in the sentence. For the entropy, we also strip stop-words, leaving only words that convey meaning and calculate the information entropy according to:

$$\mathsf{H}(X) = -p(X) \cdot log_2(p(X)). \tag{5.4}$$

We calculated both metrics for the three sentences and present them in Table 9. From those results, we see that indeed, students seem to prefer higher quantity of information even if sentence length is larger ($\Xi$), but solely length and entropy are not enough to represent their preference in this case. We see that, while explanation $K$ was the least

Table 9: Descriptive measures for sentences $\Xi$, $\Gamma$ and $K$. Source: Author.

| **Format** | **Preference** | **Length** | **Entropy** | **Info density** |
|---|---|---|---|---|
| $\Gamma$ | 27.4% | 25 words | 3.55 bits | 0.142 bits/word |
| $K$ | 22.6% | 33 words | 4.04 bits | 0.122 bits/word |
| $\Xi$ | 50.0% | 40 words | 4.06 bits | 0.102 bits/word |

preferred, it displays a higher quantity of information (entropy) than $\Gamma$.

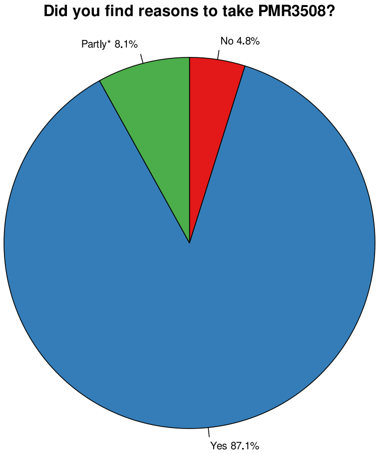
Hence, another factor we may look at is the information relative to the length for each sentence. The intuition is that this could be the preference indicator as it carries a density of information, requiring less effort to convey the same level of information to the readers. This measure is also present in Table 9, but we see that it follows the same pattern as before. For our three formats, explanations with higher sentence length also exhibit higher entropy and higher information density.

For the question of explanation format preference, we conclude that user preference is often more complex than it can be described by simple metrics. This only emphasizes the importance of carrying out studies with real users and asking questions to elicit, confirm and abide by their preferences.

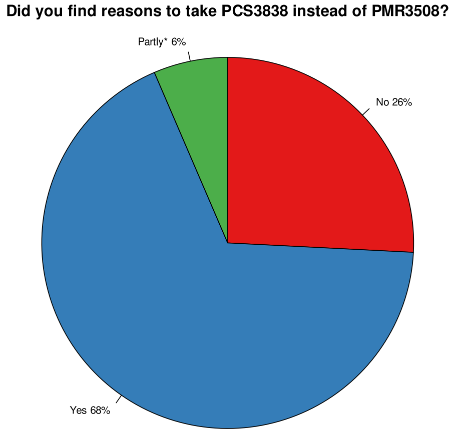
We also took the opportunity to ask clarifying questions regarding the balancing effect of presenting both reasons for and reasons against a recommendation. For that, we asked students two more questions; we present results in Figure 22. First, we asked whether they could find reasons to take the first recommended class. We see in Figure 22a that most students (87%) found reasons to take the class, while 8% reported to partly find a reason and about 5% reported not finding any reason to take the course.

Finally, we asked them whether they could find the reasons against only following the first recommendation (PMR3508) in favor of PCS3838. In Figure 22b, we see that 68% of the students reported to be able to find a reason this time. To understand why about 26% of the students could not find a reason to take PCS3838 instead of PMR3508, we found that it does not mean they did not see reasons against the first recommendation. Instead, some found did not find the reason convincing enough for their objective. For instance, one student stated that "It depends on my interest in complex systems ...", thinking that the argument against PMR3508 was that PCS3838 had more emphasis on the subject of complex systems. So, this shows that they did recognize a reason not to follow directly the recommendation, but did not necessarily consider it to be a defining factor.

Taking into account the survey results we discussed in this section, we conclude that, overall, students prefer to have more information to decide. That is how we interpret the




(a) Reasons to take PMR3508.




(b) Reasons to take PCS3838 instead.

Figure 22: Reasons for and reasons against PMR3508. Source: Author.

results from Figure 21. Furthermore, students were also able to recognize reasons for and reasons against taking the recommendation as seen in Figure 22.

# 6 CONCLUSION

*"The more I know, the more I realize I know nothing."*

-- Socrates

This master's thesis presented methods for generating explanations for automatically-generated recommendations. It also described the implementation of a recommender system that uses those methods to explain recommendations in the education domain. In addition, it shows that the proposed methods are feasible in practice in terms of coverage, support and time complexity; they lead to higher transparency and user trust when compared to the baseline algorithms that only present reasons supporting the recommendation.

We call this class of explanation methods *responsible* as they show not only the reasons to do what the recommender suggests, but also reasons for taking other paths. Hence the algorithms address a trade-off between the interests of the recommendation provider and the users of the recommendation. This lets users choose what is best for them.

Taking into account the social nature of explaining automatically-generated recommendations to humans, we carried out a survey with users to assess their preferences for explanation formats. We also investigated their perceptions of reasons for and reasons against in our explanations. We found that user preferences are more complex than what can be represented by simple sentence characteristics such as length. With respect to the presence of reasons for and reasons against, we found that most users could identify them in the explanation.

The main contributions of this work are the techniques that generate reasons for and reasons against recommendations using a knowledge graph. Scheme S1 allows one to understand further available options and their advantages and disadvantages. Scheme S5 offers explanations that display the fitness between the recommendation and user preferences.

We published a paper [61], and participated as co-first author in a second paper under review by an international journal, during this work.

Future work should improve explanations in three directions. First, it is desirable to improve the performance of the cognitive phase of the explanation process, by improving the embedding training with data enrichment and embedding types such as [79]. Second, there should be an effort to investigate how much explanations that do not shed light onto the recommendation contribute to understand that the recommendation itself might have come from spurious correlations. Third, future work should look for better ways to format explanations with reasons for and against. This could be done by gathering user feedback on which explanations are helpful, for example, using a freely-available interface. Among the various hypotheses to be tested, one for instance might vary sentence length and wording.

# REFERENCES

[1] Abu-Mostafa, Yaser S, Malik Magdon-Ismail, and Hsuan-Tien Lin. *Learning From Data*. Vol. 4. Lectures available at http://work.caltech.edu/telecourse.html. AMLBook New York, NY, USA: 2012. ISBN: 1600490069.

[2] Alharbi, Basma. "Back to Basics: An Interpretable Multi-Class Grade Prediction Framework". In: *Arabian Journal for Science and Engineering* (2021), pp. 1–16. URL: `https://doi.org/10.1007/s13369-021-06153-x`.

[3] Alshammari, M., O. Nasraoui, and S. Sanders. "Mining Semantic Knowledge Graphs to Add Explainability to Black Box Recommender Systems". In: *IEEE Access* 7 (2019), pp. 110563–110579. DOI: `10.1109/ACCESS.2019.2934633`.

[4] Alvarez, Maria. *Reasons for Action: Justification, Motivation, Explanation*. URL: `https://plato.stanford.edu/entries/reasons-just-vs-expl/#NormReas`.

[5] Amgoud, Leila et al. "On bipolarity in argumentation frameworks". In: *International Journal of Intelligent Systems* 23.10 (2008), pp. 1062–1093.

[6] Anand, Sarabjot Singh and Bamshad Mobasher. "Intelligent Techniques for Web Personalization". In: *ITWP'03: Proceedings of the 2003 international conference on Intelligent Techniques for Web Personalization*. ITWP'03. Acapulco, Mexico: Springer-Verlag, 2003, pp. 1–36. ISBN: 3540298460. DOI: `10.1007/11577935_1`. URL: `https://doi.org/10.1007/11577935%5C_1`.

[7] Army, United States Government US. *A Procedure for Requirements Analysis Systems Engineering Fundamentals (PDF)*. [Online] Access on 06 dec 2020. Archive.org, 2001. ISBN: 978-1484120835. URL: `https://web.archive.org/web/20170131231503/http://www.dau.mil/publications/publicationsdocs/sefguide%2001-01.pdf`.

[8] Atkinson, Paul. "ENIAC versus Colossus and the early presentation of electronic computers". Accessed on December 24th. 2021. Sept. 2014. URL: `http://shura.shu.ac.uk/9501/`.

[9] Bishop, Christopher M. *Pattern Recognition and Machine Learning*. Berlin, Heidelberg: Springer-Verlag, 2006. ISBN: 0387310738.

[10] Bland, J Martin and Douglas G Altman. "Statistics Notes: Measurement error". In: *BMJ* 313.7059 (1996), p. 744. ISSN: 0959-8138. DOI: 10.1136/bmj.313.7059.744. eprint: https://www.bmj.com/content. URL: https://www.bmj.com/content/313/7059/744.1.

[11] Bordes, Antoine et al. "Translating Embeddings for Modeling Multi-Relational Data". In: *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*. NIPS'13. Lake Tahoe, Nevada: Curran Associates Inc., 2013, pp. 2787–2795. URL: https://proceedings.neurips.cc/paper/2013/file/1cecc7a77928ca8133fa24680a88d2f9-Paper.pdf.

[12] Carbonell, Jaime G, Ryszard S Michalski, and Tom M Mitchell. "An overview of machine learning". In: *Machine learning*. Elsevier, 1983, pp. 3–23.

[13] Card, Stuart K., Allen Newell, and Thomas P. Moran. *The Psychology of Human-Computer Interaction*. USA: L. Erlbaum Associates Inc., 1983. ISBN: 0898592437.

[14] Cayrol, Claudette and Marie-Christine Lagasquie-Schiex. "On the Acceptability of Arguments in Bipolar Argumentation Frameworks". In: *European Conference on Symbolic and Quantitative Approaches to Reasoning and Uncertainty*. Springer. 2005, pp. 378–389.

[15] Chen, Li and Pearl Pu. "Critiquing-based recommenders: survey and emerging trends". In: *User Modeling and User-Adapted Interaction* 22.1 (2012), pp. 125–150.

[16] Chen, Li and Pearl Pu. "Evaluating critiquing-based recommender agents". In: *AAAI*. Vol. 6. 2006, pp. 157–162.

[17] Cormen, Thomas H. et al. *Introduction to Algorithms*. 2nd. The MIT Press, 2001. ISBN: 0262032937. URL: http://www.amazon.com/Introduction-Algorithms-Thomas-H-Cormen.

[18] Cormode, Graham and Balachander Krishnamurthy. "Key differences between Web 1.0 and Web 2.0". In: *First Monday* (2008).

[19] Corrêa, André Murillo Geraldo and Fabio Gagliardi Cozman. *Jupiterweb Chatbot*. 2018. URL: https://bdta.aguia.usp.br/item/002954978.

[20] Doshi-Velez, Finale and Been Kim. *Towards A Rigorous Science of Interpretable Machine Learning*. cite arxiv:1702.08608. 2017. URL: http://arxiv.org/abs/1702.08608.

[21] Estola, Evan. “When Recommendation Systems Go Bad”. In: *Proceedings of the 10th ACM Conference on Recommender Systems*. RecSys ’16. Boston, Massachusetts, USA: Association for Computing Machinery, 2016, p. 367. ISBN: 9781450340359. DOI: 10.1145/2959100.2959117. URL: https://doi.org/10.1145/2959100.2959117.

[22] Freiberger, Paul A. and Michael R. Swaine. *ENIAC*. Accessed on December 24th. 2021. May 27, 2020. URL: https://www.britannica.com/technology/ENIAC.

[23] Friedenthal, Sanford, Alan Moore, and Rick Steiner. “Chapter 4 - An Automobile Example Using the SysML Basic Feature Set”. In: *A Practical Guide to SysML (Third Edition)*. Ed. by Friedenthal, Sanford, Alan Moore, and Rick Steiner. Third. The MK/OMG Press. Boston: Morgan Kaufmann, 2015, pp. 53–81. ISBN: 978-0-12-800202-5. DOI: https://doi.org/10.1016/B978-0-12-800202-5.00004-7. URL: https://www.sciencedirect.com/science/article/pii/B9780128002025000047.

[24] Gardner, Matt and Tom Mitchell. “Efficient and Expressive Knowledge Base Completion Using Subgraph Feature Extraction”. In: *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 2015, pp. 1488–1498. DOI: 10.18653/v1/D15-1173.

[25] Goodman, Bryce and Seth Flaxman. “European Union Regulations on Algorithmic Decision-Making and a “Right to Explanation””. In: *AI Magazine* 38.3 (Oct. 2017), pp. 50–57. ISSN: 0738-4602. DOI: 10.1609/aimag.v38i3.2741. URL: http://dx.doi.org/10.1609/aimag.v38i3.2741.

[26] Gunning, David and David Aha. “DARPA’s Explainable Artificial Intelligence (XAI) Program”. In: *AI Magazine* 40.2 (June 2019), pp. 44–58. DOI: 10.1609/aimag.v40i2.2850. URL: https://www.aaai.org/ojs/index.php/aimagazine/article/view/2850.

[27] Hägglund, Sture. “Introducing expert critiquing systems”. In: *The Knowledge Engineering Review* 8.4 (1993), pp. 281–284. DOI: 10.1017/S0269888900000308.

[28] He, Ruining, Wang-Cheng Kang, and Julian McAuley. “Translation-Based Recommendation”. In: *Proceedings of the Eleventh ACM Conference on Recommender Systems*. RecSys ’17. Como, Italy: Association for Computing Machinery, 2017, pp. 161–169. ISBN: 9781450346528. DOI: 10.1145/3109859.3109882. URL: https://doi.org/10.1145/3109859.3109882.

[29] Henderson, Anthony. *Introduction to Graph Theory.* Distributed by the School of Mathematics and Statistics, The University of Sydney. Mar. 2014.

[30] Henk, Veronika et al. *Metaresearch Recommendations using Knowledge Graph Embeddings.* 2018.

[31] Hoeve, Maartje ter et al. "Faithfully Explaining Rankings in a News Recommender System". In: *CoRR* abs/1805.05447 (2018). arXiv: 1805.05447. URL: http://arxiv.org/abs/1805.05447.

[32] Hogan, Aidan et al. *Knowledge Graphs.* English. Synthesis Lectures on Data, Semantics, and Knowledge 22. Morgan & Claypool, 2021. ISBN: 9781636392363. DOI: 10.2200/S01125ED1V01Y202109DSK022. URL: https://kgbook.org/.

[33] Huang, Xiao et al. "Knowledge Graph Embedding Based Question Answering". In: *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining.* WSDM '19. Melbourne VIC, Australia: Association for Computing Machinery, 2019, pp. 105–113. ISBN: 9781450359405. DOI: 10.1145/3289600.3290956. URL: https://doi.org/10.1145/3289600.3290956.

[34] Jannach, Dietmar et al. "A Survey on Conversational Recommender Systems". In: *ACM Computing Surveys* 54.5 (May 2021). ISSN: 0360-0300. DOI: 10.1145/3453154. URL: https://doi.org/10.1145/3453154.

[35] Jumper, John et al. "High Accuracy Protein Structure Prediction Using Deep Learning". In: *Fourteenth Critical Assessment of Techniques for Protein Structure Prediction (Abstract Book).* Nov. 2020.

[36] Keeney, R.L., H. Raiffa, and R.F. Meyer. *Decisions with Multiple Objectives: Preferences and Value Trade-Offs.* Wiley series in probability and mathematical statistics. Applied probability and statistics. Cambridge University Press, 1993. ISBN: 9780521438834. URL: https://books.google.de/books?id=GPE6ZAqGrnoC.

[37] Koren, Yehuda, Robert Bell, and Chris Volinsky. "Matrix Factorization Techniques for Recommender Systems". In: *Computer* 42.8 (Aug. 2009), pp. 30–37. ISSN: 0018-9162. DOI: 10.1109/MC.2009.263. URL: https://doi.org/10.1109/MC.2009.263.

[38] Lao, Ni, Tom Mitchell, and William W. Cohen. "Random Walk Inference and Learning in A Large Scale Knowledge Base". In: *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing.* Edinburgh, Scotland, UK.: Association for Computational Linguistics, July 2011, pp. 529–539. URL: https://aclanthology.org/D11-1049.

[39] Larrañaga, Pedro et al. *Industrial Applications of Machine Learning.* CRC Press, 2018.

[40] Likert, Rensis. "A Technique for the Measurement of Attitudes". In: *Archives of Psychology* 140 (1932), pp. 1–55.

[41] Ma, Hao et al. "Recommender Systems with Social Regularization". In: *Proceedings of the Fourth ACM International Conference on Web Search and Data Mining*. WSDM '11. Hong Kong, China: Association for Computing Machinery, 2011, pp. 287–296. ISBN: 9781450304931. DOI: 10.1145/1935826.1935877. URL: https://doi.org/10.1145/1935826.1935877.

[42] Mahdavinejad, Mohammad Saeid et al. "Machine learning for Internet of Things Data Analysis: A Survey". In: *Digital Communications and Networks* 4.3 (2018), pp. 161–175.

[43] Markman, A.B. *Knowledge Representation.* Taylor & Francis, 2013. ISBN: 9781134802906. URL: https://books.google.com.br/books?id=Uu%5C_mxqQal8kC.

[44] Massa, Paolo and Paolo Avesani. "Trust-Aware Recommender Systems". In: *Proceedings of the 2007 ACM Conference on Recommender Systems*. RecSys '07. Minneapolis, MN, USA: Association for Computing Machinery, 2007, pp. 17–24. ISBN: 9781595937308. DOI: 10.1145/1297231.1297235. URL: https://doi.org/10.1145/1297231.1297235.

[45] Max Roser, Hannah Ritchie and Esteban Ortiz-Ospina. "Internet". In: *Our World in Data* (2015). https://ourworldindata.org/internet.

[46] Miller, Tim. "Explanation in Artificial Intelligence: Insights from the Social Sciences". In: *CoRR* abs/1706.07269 (2017). arXiv: 1706.07269. URL: http://arxiv.org/abs/1706.07269.

[47] Mohamed, Sameh K., Emir Muñoz, and Vit Novacek. "On Training Knowledge Graph Embedding Models". In: *Information* 12.4 (2021). ISSN: 2078-2489. DOI: 10.3390/info12040147. URL: https://www.mdpi.com/2078-2489/12/4/147.

[48] Molnar, Christoph. *Interpretable Machine Learning. A Guide for Making Black Box Models Explainable.* https://christophm.github.io/interpretable-ml-book/. [Online], 2019.

[49] Musto, Cataldo et al. "Linked open data-based explanations for transparent recommender systems". In: *International Journal of Human-Computer Studies* 121 (2019). Advances in Computer-Human Interaction for Recommender Systems, pp. 93–107. ISSN: 1071-5819. DOI: https://doi.org/10.1016/j.ijhcs.2018.03.003. URL: http://www.sciencedirect.com/science/article/pii/S1071581918300946.

[50] Nguyen, Dat Quoc et al. "Neighborhood Mixture Model for Knowledge Base Completion". In: *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*. Berlin, Germany: Association for Computational Linguistics, Aug. 2016, pp. 40–50. DOI: 10.18653/v1/K16-1005. URL: https://www.aclweb.org/anthology/K16-1005.

[51] Nickel, Maximilian et al. *A Review of Relational Machine Learning for Knowledge Graphs*. cite arxiv:1503.00759Comment: To appear in Proceedings of the IEEE. 2015. DOI: 10.1109/JPROC.2015.2483592. URL: http://arxiv.org/abs/1503.00759.

[52] Nielsen, J. *Usability Engineering*. Interactive Technologies. Elsevier Science, 1994. ISBN: 9780125184069. URL: https://books.google.com.br/books?id=95As2OF67f0C.

[53] Norton, John. "'Nature is the Realisation of the Simplest Conceivable Mathematical Ideas': Einstein and the Canon of Mathematical Simplicity". In: *Studies In History and Philosophy of Science Part B: Studies In History and Philosophy of Modern Physics* 31 (June 2000), pp. 135–170. DOI: 10.1016/S1355-2198(99)00035-0.

[54] O'Donovan, John and Barry Smyth. "Trust in Recommender Systems". In: *Proceedings of the 10th International Conference on Intelligent User Interfaces*. IUI '05. San Diego, California, USA: Association for Computing Machinery, 2005, pp. 167–174. ISBN: 1581138946. DOI: 10.1145/1040830.1040870. URL: https://doi.org/10.1145/1040830.1040870.

[55] Parliament, European. *REGULATION (EU) 2016/679 OF THE EUROPEAN PARLIAMENT AND OF THE COUNCIL of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation)*. URL: http://data.europa.eu/eli/reg/2016/679/oj.

[56] Polleti, Gustavo Padilha. "Explanation Generation For Conversational Recommendation Systems Based On Knowledge Embeddings". MA thesis. Polytechnic School, University of São Paulo, 2022.

[57] Polleti, Gustavo Padilha and Fabio Gagliardi Cozman. *Chatbot para Perguntas e Respostas com Capacidade de Explicação*. 2019.

[58] Polleti, Gustavo Padilha and Fabio Gagliardi Cozman. "Explaining Content-Based Recommendations with Topic Models". In: *2019 8th Brazilian Conference on Intelligent Systems (BRACIS)*. Oct. 2019, pp. 800–805. DOI: `10.1109/BRACIS.2019.00143`.

[59] Polleti, Gustavo Padilha and Fabio Gagliardi Cozman. "Faithfully explaining predictions of knowledge embeddings". In: *Encontro Nacional de Inteligência Artificial (ENIAC)*. Sept. 2019, pp. 1–12.

[60] Polleti, Gustavo Padilha, Hugo Neri Munhoz, and Fabio Gagliardi Cozman. "Explanations within Conversational Recommendation Systems: Improving Coverage through Knowledge Graph Embedding". In: *2020 AAAI Workshop on Interactive and Conversational Recommendation System*. New York City, New York, USA: AAAI Press, 2020, p. 8.

[61] Polleti, Gustavo Padilha, Douglas Luan de Souza, and Fabio Cozman. "Why should I not follow you? Reasons For and Reasons Against in Responsible Recommender Systems". In: *3rd FAccTRec Workshop: Responsible Recommendation*. 2020, p. 6. arXiv: `2009.01953 [cs.AI]`.

[62] Pressman, Roger. *Software Engineering: A Practitioner's Approach*. 7th ed. USA: McGraw-Hill, Inc., 2009. ISBN: 0073375977.

[63] Ribeiro, Marco Tulio, Sameer Singh, and Carlos Guestrin. "'Why Should I Trust You?': Explaining the Predictions of Any Classifier". In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*. 2016, pp. 1135–1144.

[64] Ribeiro, Marco Tulio, Sameer Singh, and Carlos Guestrin. *Model-Agnostic Interpretability of Machine Learning*. June 2016. arXiv: `1606.05386.pdf`. URL: `http://arxiv.org/abs/1606.05386.pdf`.

[65] Ricci, Francesco et al. *Recommender Systems Handbook*. 1st. Berlin, Heidelberg: Springer-Verlag, 2010. ISBN: 0387858199.

[66] Russell, Stuart. "Should we fear supersmart robots?" In: *Scientific American* 314.6 (2016), pp. 58–59.

[67] Russell, Stuart and Peter Norvig. *Artificial Intelligence: A Modern Approach*. 3rd. USA: Prentice Hall Press, 2009. ISBN: 0136042597.

[68] Shani, Guy and Asela Gunawardana. "Evaluating Recommendation Systems". In: *Recommender Systems Handbook*. Boston, MA: Springer US, 2011, pp. 257–297. ISBN: 978-0-387-85820-3. DOI: 10.1007/978-0-387-85820-3_8. URL: https://doi.org/10.1007/978-0-387-85820-3%5C_8.

[69] Smith, Brent and Greg Linden. "Two Decades of Recommender Systems at Amazon.com". In: *IEEE Internet Computing* 21.3 (2017), pp. 12–18. DOI: 10.1109/MIC.2017.72.

[70] Snedegar, J. *Contrastive Reasons*. OUP Oxford, 2017. ISBN: 9780191089039. URL: https://books.google.com.br/books?id=9zJdDgAAQBAJ.

[71] Snedegar, Justin. "Reasons for and reasons against". In: *Philosophical Studies* 175.3 (2018), pp. 725–743.

[72] *SysML FAQ: What is a Requirement Diagram (REQ) and how is it used?* Accessed: 2023-04-22. URL: https://sysml.org/sysml-faq/what-is-requirement-diagram.html.

[73] Tintarev, Nava and Judith Masthoff. "A Survey of Explanations in Recommender Systems". In: *Proceedings of the 2007 IEEE 23rd International Conference on Data Engineering Workshop*. ICDEW '07. Washington, DC, USA: IEEE Computer Society, 2007, pp. 801–810. ISBN: 978-1-4244-0831-3. DOI: 10.1109/ICDEW.2007.4401070. URL: http://dx.doi.org/10.1109/ICDEW.2007.4401070.

[74] Valletta, John Joseph et al. "Applications of machine learning in animal behaviour studies". In: *Animal Behaviour* 124 (2017), pp. 203–220.

[75] Vamathevan, Jessica et al. "Applications of machine learning in drug discovery and development". In: *Nature Reviews Drug Discovery* 18.6 (2019), pp. 463–477.

[76] Walsh, Dorothy. "Occam's razor: A principle of intellectual elegance". In: *American Philosophical Quarterly* 16.3 (1979), pp. 241–244.

[77] Wang, Q. et al. "Knowledge Graph Embedding: A Survey of Approaches and Applications". In: *IEEE Transactions on Knowledge and Data Engineering* 29.12 (2017), pp. 2724–2743. DOI: 10.1109/TKDE.2017.2754499.

[78] Wohlin, Claes et al. *Experimentation in Software Engineering*. Springer Publishing Company, Incorporated, 2012. ISBN: 3642290434.

[79] Yao, Zhen et al. *Analogical Inference Enhanced Knowledge Graph Embedding*. 2023. arXiv: 2301.00982 [cs.AI].

[80] Zhang, Jiyong, Nicolas Jones, and Pearl Pu. "A visual interface for critiquing-based recommender systems". In: *Proceedings of the 9th ACM conference on Electronic commerce*. 2008, pp. 230–239.

[81] Zhang, Jiyong and Pearl Pu. "A comparative study of compound critique generation in conversational recommender systems". In: *International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems*. Springer. 2006, pp. 234–243.

[82] Zhang, Min and Yiqun Liu. "A commentary of TikTok recommendation algorithms in MIT Technology Review 2021". In: *Fundamental Research* 1.6 (2021), pp. 846–847.

[83] Zhang, Wen et al. "Interaction Embeddings for Prediction and Explanation in Knowledge Graphs". In: *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining* (Jan. 2019). DOI: 10.1145/3289600.3291014. URL: http://dx.doi.org/10.1145/3289600.3291014.

[84] Zhang, Yongfeng and Xu Chen. "Explainable Recommendation: A Survey and New Perspectives". In: *CoRR* abs/1804.11192 (2018), pp. 1–101. arXiv: 1804.11192. URL: http://arxiv.org/abs/1804.11192.

# APPENDIX A – THE IMPLEMENTED CRS

This appendix shows screenshots from interactions with our implemented system. It is worth noting that dialogues are in Brazilian Portuguese. However, conversations are transcribed and translated.

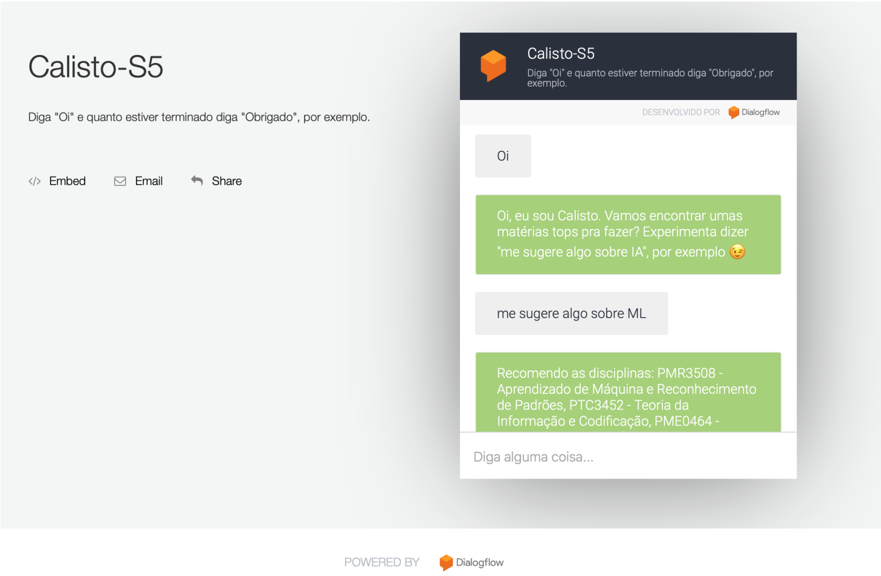
Figure 23a depicts the entire page where Calisto is available in a version that implements Snedegar's Scheme 5 (S5). The dialogue starts with a greeting. Calisto replies and gives some help text; the user asks for a recommendation about ML, which is correctly understood to mean Machine Learning and some recommendations are given back. In Figure 23b the chat continues, and explanations are requested for two different subjects - PMR3508 and PTC3452 - to which intuitive and correct textual explanations are given back. PMR3508 is related to probabilistic classifiers, and PTC3452 is related to non-supervised learning. Both topics belong to the category of machine learning and those are the reasons for recommending those units of studies for the theme ML.

The back-end allows for debugging and monitoring through logs sent to the docker container with the terminal command `docker logs -f calisto-s5`. Figure 24 shows the resulting screen with the HTTP requests received with URL and method, as well as the response with JSON explanation with response code.

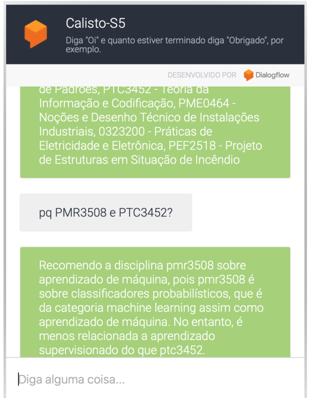
The deployment used a docker container to provide environment isolation to the system in production. The codes in Listing 1 and Listing 2 show the commands for creating the docker image and running it, respectively. The figure is based on Python 3.7.9 and includes Node.js for Localtunnel [1], Localtunnel itself for reverse proxying/tunnelling, Calisto's back-end. For running Calisto, we set it to restart on failure, mirror the port in the docker container to the host machine's port and set the Nvidia devices to access GPUs.

All the experiments were run using the computational infrastructure at the Centro de Inteligência Artificial e Aprendizado de Máquina (CIAAM) hosted by USP Innovation Center (InovaUSP).

[1] https://theboroer.github.io/localtunnel-www/




(a) Greeting, recommendation request and response.




(b) Example response for an explanation request.

Figure 23: Calisto chatbot interface. Source: Author.

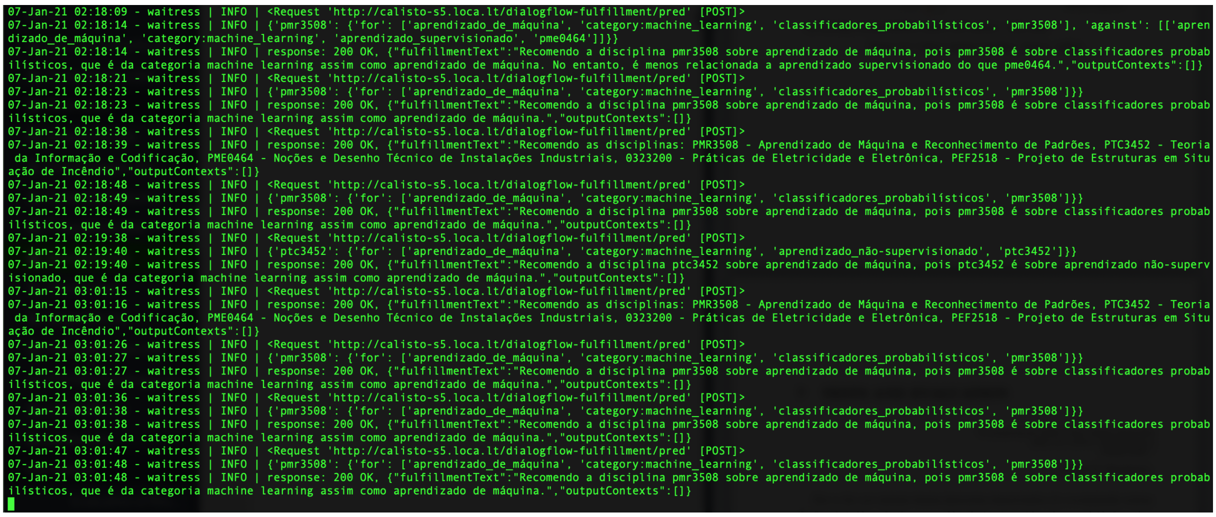



Figure 24: Calisto's back-end running (logs). Source: Author.

```
FROM python:3.7.9
RUN pip install --upgrade pip
RUN mkdir -p calisto-backend
ADD . calisto-backend
WORKDIR calisto-backend
RUN pip install -r requirements.txt
RUN curl https://nodejs.org/dist/v12.19.0/node-v12.19.0-linux-x64.tar.xz
 ↪  --output node-v12.19.0-linux-x64.tar.xz
RUN mkdir -p /usr/local/lib/nodejs
RUN tar -xJvf node-v12.19.0-linux-x64.tar.xz -C /usr/local/lib/nodejs
ENV PATH=/usr/local/lib/nodejs/node-v12.19.0-linux-x64/bin:$PATH
RUN npm install -g localtunnel
ENV PORT=5024
CMD while true; do lt -s calisto-s5 -p $PORT ; sleep 1; done & python
 ↪  src/server.py
```

Listing 1: Code for Dockerfile.

```
docker build . -t calisto-backend-gpu
docker run --device=/dev/nvidia0 --device=/dev/nvidia1 -p 5023:5023
 ↪  --restart on-failure --name calisto-gpu -tid calisto-backend-gpu
```

Listing 2: Code for back-end deployment.

# APPENDIX B – EXPERIMENTAL SETUP

In this chapter we present setups used to carry experiments we explained in Chapter 5. Please note that screenshots are in Brazilian Portuguese, as experiments were carried out in Brazil.

## B.1 Experiment with Users and Real System


Figure 25: Experiment platform (top of the page). Source: Author.

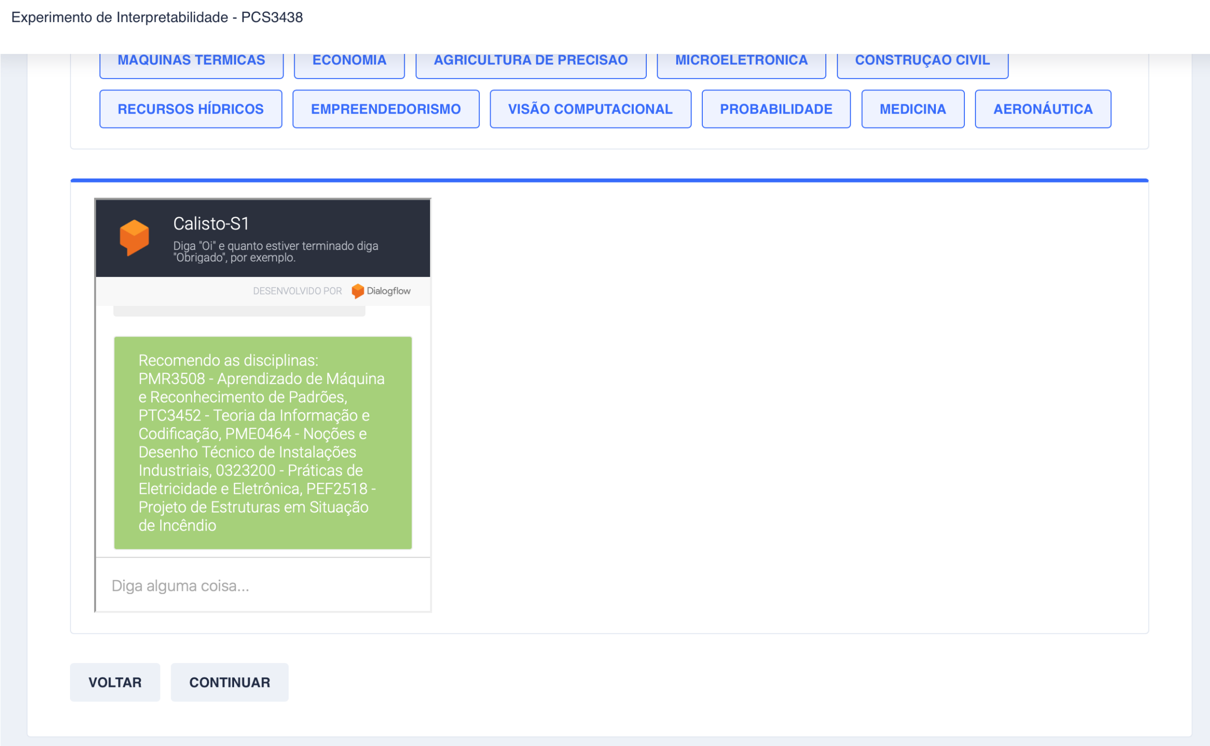



Figure 26: Experiment platform (bottom of the page). Source: Author.

## B.2 Experiment with Users for Explanation Format

We list scripts we used for the analysis of our experiment with users to investigate their preferences for each explanation format and also present the questionnaire for it.

```
import numpy as np
import nltk
from nltk.corpus import stopwords
import pandas as pd
from nltk.stem import WordNetLemmatizer

lemmatizer = WordNetLemmatizer()
nltk.download('stopwords')
nltk.download('wordnet')
stop_words = set(stopwords.words('english'))

csi = "I recommend the course COURSE1 as the most adherent to TOPIC1,
↪  since COURSE1 is about TOPIC2 from the category CATEGORY as TOPIC1.
↪  However, it covers the subject SUBJECT less than COURSE2, which is
↪  also about TOPIC1."

gamma = "I recommend COURSE1 as the most adherent to TOPIC1. However, it
↪  covers the subject SUBJECT less than COURSE2, which is also about
↪  TOPIC1."

kappa = "I recommend COURSE1 about TOPIC1, since it is about TOPIC2 from
↪  the category CATEGORY as TOPIC1. However, COURSE1 covers the subject
↪  SUBJECT less than COURSE2."

def unique_counter(sentence):
    tokens = nltk.word_tokenize(sentence)
    words = [token for token in tokens if not token in stop_words and
    ↪  token.isalnum()]
    words = [lemmatizer.lemmatize(word) for word in words]
    return len(words)
```

```
def length(sentence):
    tokens = nltk.word_tokenize(sentence)
    words = [token for token in tokens if token.isalnum()]
    return len(words)

def entropy(sentence, base=2):
    tokens = nltk.word_tokenize(sentence)
    words = [token for token in tokens if not token in stop_words and
    ↪   token.isalnum()]
    unique_words = set(words)
    n = len(words)
    frequencies = [words.count(word)/n for word in unique_words]
    base_factor = np.log(base)
    return -(frequencies * np.log(frequencies)/base_factor).sum()
sentences_df = pd.DataFrame([['Csi', length(csi), unique_counter(csi),
↪   entropy(csi)], ['Gamma', length(gamma), unique_counter(gamma),
↪   entropy(gamma)], ['Kappa', length(kappa), unique_counter(kappa),
↪   entropy(kappa)]], columns=['Name', 'Length', 'Unique', 'Entropy'])
sentences_df['Relative Info'] =
↪   sentences_df['Entropy']/sentences_df['Length']
sentences_df['Relative Unique'] =
↪   sentences_df['Entropy']/sentences_df['Unique']
sentences_df['User Pref'] = pd.Series([50, 27.4, 22.6])
sentences_df
```

# Explicações sobre Recomendações

Neste formulário, serão apresentadas explicações sobre recomendações de disciplinas

* Indica uma pergunta obrigatória

1. As respostas fornecidas nesta atividade serão utilizadas de forma anônima e apenas para propósitos acadêmicos. Nenhuma informação será compartilhada ou aberta para terceiros fora do âmbito desta pesquisa. Você concorda com esses termos? *

   *Marcar apenas uma oval.*

   - Sim

*Pular para a pergunta 2*

## Informações Cadastrais

Suas respostas serão usadas para traçar o perfil de respondentes para fins estatísticos, mas de forma anônima.

2. Qual a sua idade? *

   ______________________________

3. Qual seu número USP *

   Apenas para fins de atribuição de nota, para saber quem participou.

   ______________________________

4. Com qual gênero você se identifica? *

   *Marcar apenas uma oval.*

   - Masculino
   - Feminino
   - Não-binário
   - Não desejo responder

5. Qual a sua região de origem? *

*Marcar apenas uma oval.*

- Norte
- Nordeste
- Sudeste
- Centro-oeste
- Sul
- Outro País de Língua Portuguesa
- Outros Países

6. O quanto você concorda com a afirmação "Uso computador ou smartphone diariamente e tenho total afinidade com novas tecnologias"

*Marcar apenas uma oval.*

Discordo totalmente

1

2

3

4

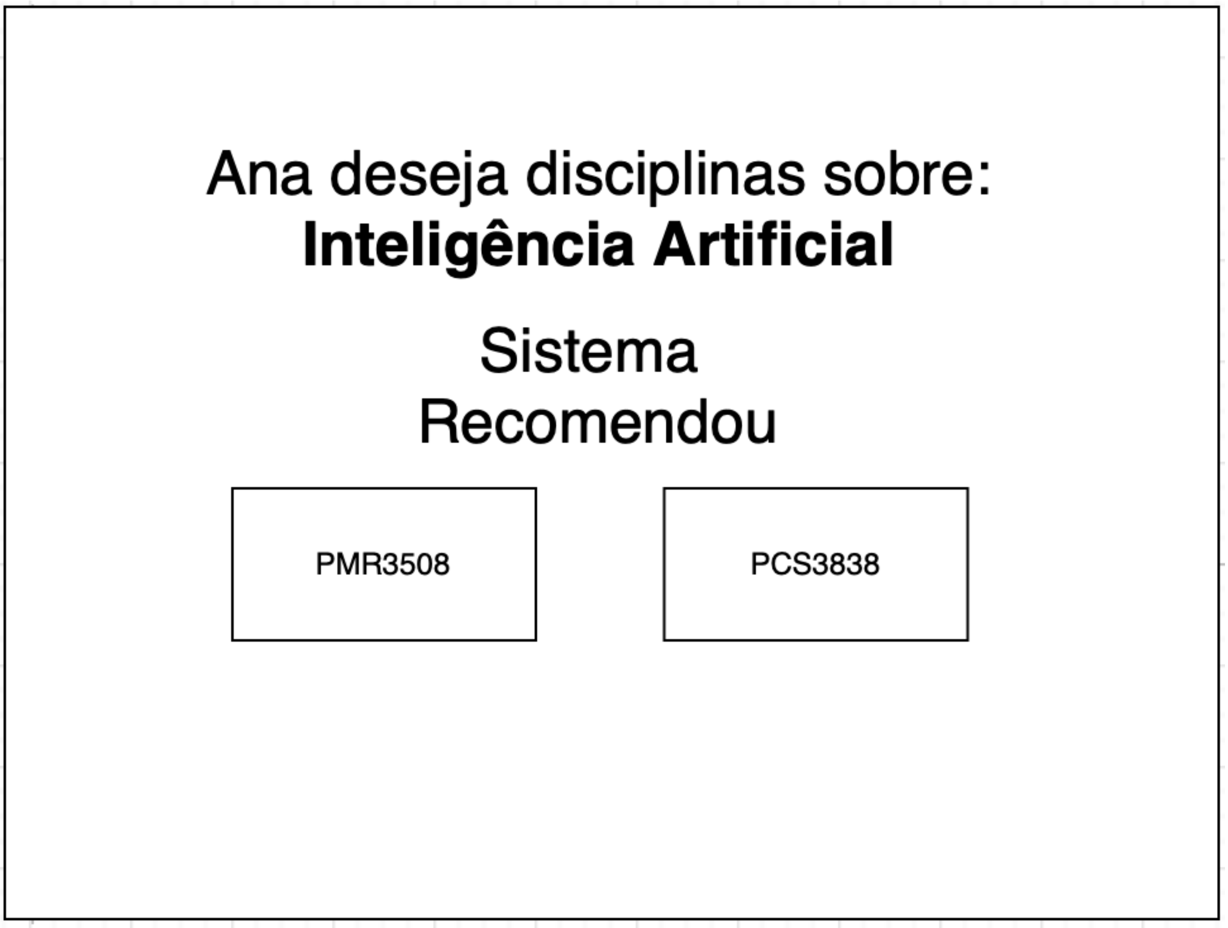
5

Concordo totalmente

## Explicações sobre Recomendações

Ana é estudante de graduação de engenharia de computação e é entusiasta de Inteligência Artificial. Para aprender mais sobre o tema, ela pede a ajuda de um sistema de recomendação de disciplinas da USP.

Ana, ao pedir por disciplinas sobre Inteligência Artificial, recebe as seguintes recomendações: PMR3508 e PCS3838. Para entender melhor qual disciplina deve escolher, ela pede por explicações.




Formatos de Explicação

As seguintes explicações foram apresentadas a Ana. Avalie os formatos de explicação abaixo, para as mesmas recomendações.

7. Explicação Gama: Em uma escala de 1-5, diga o quanto você concorda com a seguinte afirmação: "Eu sinto que a explicação abaixo me ajudou/foi útil" *

Recomendo PMR3508 como a mais aderente a inteligência artificial. No entanto, ela cobre menos o assunto de sistemas complexos do que PCS3838, que também é sobre inteligência artificial.

*Marcar apenas uma oval.*

Discordo Totalmente

1 ◯

2 ◯

3 ◯

4 ◯

5 ◯

Concordo Totalmente

8. Explicação Kapa: Em uma escala de 1-5, diga o quanto você concorda com a seguinte afirmação: "Eu sinto que a explicação abaixo me ajudou/foi útil" *

Recomendo PMR3508 sobre inteligência artificial. pois ela é sobre aprendizado de máquina, que é da categoria cibernética assim como inteligência artificial. No entanto, PMR3508 cobre menos o assunto sistemas complexos do que PCS3838.

*Marcar apenas uma oval.*

Discordo Totalmente

1 ◯

2 ◯

3 ◯

4 ◯

5 ◯

Concordo Totalmente

9. Explicação Csi: Em uma escala de 1-5, diga o quanto você concorda com a seguinte afirmação: "Eu sinto que a explicação abaixo me ajudou/foi útil" *

Recomendo a disciplina PMR3508 como a mais aderente a inteligência artificial, pois PMR3508 é sobre aprendizado de máquina, que é da categoria cibernética assim como inteligência artificial. No entanto, ela cobre menos o assunto de sistemas complexos do que PCS3838, que também é sobre inteligência artificial.

*Marcar apenas uma oval.*

Discordo Totalmente

1 ⬭

2 ⬭

3 ⬭

4 ⬭

5 ⬭

Concordo Totalmente

Esta seção se aplica para todo os formatos de explicação apresentados

10. Caso você tenha uma sugestão de como melhorar as explicações apresentadas, escreva-a abaixo:

11. Qual das explicações anteriores te ajuda melhor a escolher a disciplina? *

*Marcar apenas uma oval.*

- Explicação Gama
- Explicação Kapa
- Explicação Csi

12. Nas explicações fornecidas, você identificou alguma razão para aceitar a recomendação? Ou seja, para cursar alguma das disciplinas recomendadas. *

*Marcar apenas uma oval.*

- Sim
- Não
- Outro: ____________________

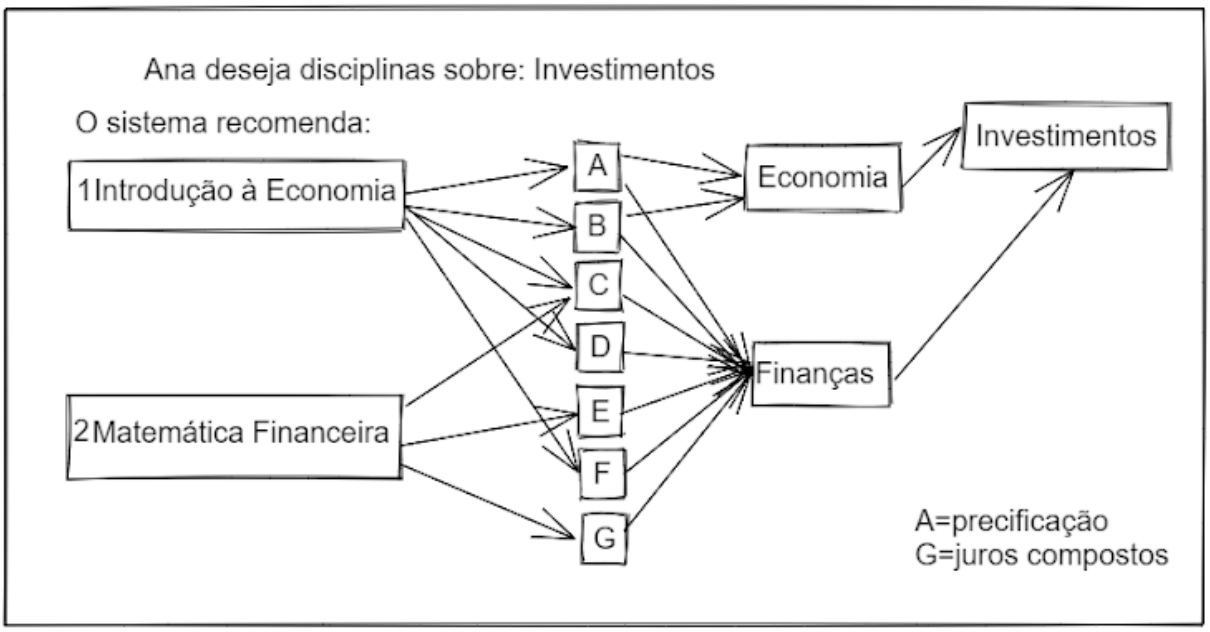
13. Nas explicações fornecidas, você identificou razões para cursar a disciplina PCS3838 ao invés de PMR3508? *

*Marcar apenas uma oval.*

- Sim
- Não
- Outro: ____________________

Gerando uma explicação

14. Dado o contexto da imagem abaixo, como você explicaria esta recomendação? *




Comentário final

15. Algum comentário adicional que você queira expressar:

# ANNEX A – RELATED PAPER - FACCTREC @ RECSYS 2020

The following article was published in the *3rd FAccTRec Workshop: Responsible Recommendation*, part of the 14th ACM Recommender Systems Conference (RecSys2020) to report on research carried out using the system described in this thesis (Calisto).

# Why should I *not* follow you? Reasons For and Reasons Against in Responsible Recommender Systems

Gustavo P. Polleti
Universidade de São Paulo
gustavo.polleti@usp.br

Douglas L. de Souza
Universidade de São Paulo
douglas.luan.souza@usp.br

Fabio G. Cozman
Universidade de São Paulo
fgcozman@usp.br

## ABSTRACT

A few Recommender Systems (RS) resort to explanations so as to enhance trust in recommendations. However, current techniques for explanation generation tend to strongly uphold the recommended products instead of presenting both reasons for and reasons against them. We argue that an RS can better enhance overall trust and transparency by frankly displaying both kinds of reasons to users. We have developed such an RS by exploiting knowledge graphs and by applying Snedegar's theory of practical reasoning. We show that our implemented RS has excellent performance and we report on an experiment with human subjects that shows the value of presenting both reasons for and against, with significant improvements in trust, engagement, and persuasion.

**ACM Reference Format:**

Gustavo P. Polleti, Douglas L. de Souza, and Fabio G. Cozman. 2020. Why should I *not* follow you? Reasons For and Reasons Against in Responsible Recommender Systems. In *RecSys '20: Proceedings of the 14th ACM Conference on Recommender Systems.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

## 1 INTRODUCTION

Human subjects find it hard to make a decision when a very large number of options is available; a Recommender System (RS) provides valuable help by selecting a small set of options that are then evaluated by the user [20]. However, even if an RS produces sensible recommendations, users may reject them if their rationale is not understood [22]. It is thus clearly desirable to have RSs that offer sensible, transparent and trustworthy recommendations; one strategy that seems particularly promising is for the RS to generate explanations that clarify the recommendations [25].

Explanations presumably enhance transparency and trust. However, explanation generation techniques now in use in RSs focus solely on advocacy for the recommended options. By describing only the benefits of those options, they may fail to offer a balanced perspective to the user, ultimately squandering overall trust. A user may be at first happy to get some positive clarification about recommended products, but if she never sees information about possible downsides, she will ultimately lose interest in the recommendations.

We argue that an RS should provide *responsible* explanations in the sense that both reasons *for* and reasons *against* explicitly escort recommendations. We take Snedegar's theory of reasons for/against [23], a philosophical theory of practical reasoning, and realize it in the context of RSs. To do so, we start with existing procedures that generate reasons *for* by analyzing paths in knowledge graphs [1, 15, 18]. We then modify such procedures so as to detect paths (or their absence) that count as reasons *against*. Snedegar's theory relies on five schemes of reasons against; we examine their computational implementation, identifying the most promising strategies. We also describe an RS we have implemented and its practical operation with reasons for/against. Additionally, we have carried out experiments with human subjects that show our approach to responsible recommendations to yield higher overall trust in generated explanations.

The paper is organized as follows. Section 2 presents some basic notions on recommender systems, explainability, transparency, and trust. In Section 3 we propose strategies to generate reasons for/against. We then present our empirical results, and offer concluding remarks in the last section.

## 2 A BIT OF BACKGROUND: RECOMMENDATIONS, TRUST, INTERPRETABILITY, EXPLANATIONS

An RS has a set of users and a set of items, usually producing a score $r(u, i)$ that captures the affinity between user $u$ and item $i$ [20]. An RS often relies on the score to rank a number $N$ of items to be presented to the user. The definition of affinity varies wildly, depending on the application domain [8, 21]. The current state of the art is to learn the affinity between users and items from past experience using latent variable models, often dependent on matrix factorization and embedding techniques [5–7]. These techniques map items to a (numeric) latent space where similar items appear near to each other, usually by optimizing distances between related objects as they are mapped [16].

Opaque models, such as the ones produced by embeddings, create obstacles to the interpretability of recommendations [3]. Here we take interpretability as the degree to which a human can understand the cause of a decision [13]. A device may be transparent in that the user can access all elements of its operation, yet its output may have low interpretability. When interpretability is low, one possible strategy is to generate explanations for the decisions. There are several techniques for explanation generation [14]; for instance, some of them investigate the sensitivity of outputs to inputs or to elements of a model — the explanation is an indication of which parts of input/model affect the output. Other techniques aim at more elaborate explanations. Some of them are dependent on a particular model; for instance, some techniques focus on neural networks, producing explanations that involve particular neurons and layers. Other techniques for explanation generation are model agnostic; that is, they only look at inputs and outputs of the model to be explained. We focus on model-agnostic explanations in this work.

*FAccTRec2020, September 26, 2020, Online*
2020. ACM ISBN 978-x-xxxx-xxxx-x/YY/MM...$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

It is commonly stated that performance and interpretability are opposing goals [19]; for instance, an accurate classifier is a complex and hard to interpret one. However, matters are more delicate in the context of RSs, as performance itself depends on trust [17], and high interpretability is bound to increase trust (when interpretation fails, existing RSs may fail in surprising ways [4]). Previous efforts have explored various ways to obtain high performance and high interpretability [10, 11, 28], in some cases generating explanations that support recommendations [1, 15, 24].

## 3 EXPLANATIONS WITH REASONS FOR AND REASONS AGAINST

Recent RSs that rely on explanations do offer useful information to the user; however, we argue that they run into a difficult balancing act [12]. This is not unlike the salesperson who always proposes products with complimentary words, as opposed to the salesperson who frankly discusses the advantages and disadvantages of products. A perceptive customer will gradually favor a salesperson who chooses sincerity over persuasion — exactly the behavior we propose for responsible RSs.

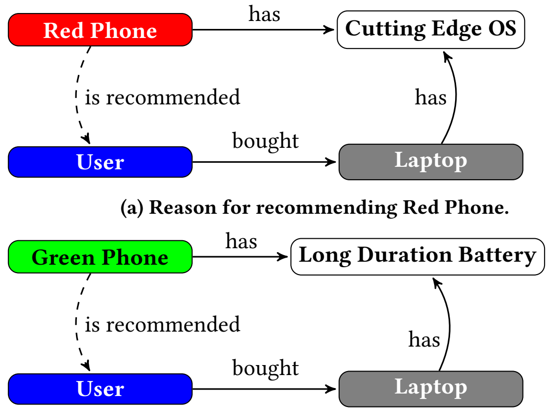
The solution, then, is to build RSs that state reasons *for* recommended items together with reasons *against* the same items. This is the main idea in this paper; to make it concrete, we first discuss techniques that generate reasons *for* (Section 3.1) and then we propose novel ideas on the generation of reasons *against* (Section 3.2).

### 3.1 Reasons For: What They Are, and How to Generate Them

Reasons for a given recommendation can be produced using an auxiliary knowledge graph (KG), a strategy that has been explored in previous efforts [1, 15, 18].

The idea is to use a KG containing all entities handled by the RS so as to find connections between users and items. A knowledge graph (KG) consists of a set of *entities* $\mathcal{E} = \{e_1, \ldots, e_{N_e}\}$ and a set of binary *relations* $\mathcal{R} = \{r_1, \ldots, r_{N_r}\}$. Using RDF notation [26], an edge in the graph can be interpreted as a triple $\langle h, r, t\rangle$ where $h$, $r$ and $t$ are, respectively, the *subject (head)*, *predicate (relation)* and *object (tail)*. The existence of a triple $x_{h,r,t} = \langle h, r, t\rangle$ is indicated by a random variable $y_{h,r,t}$ with values in $\{0, 1\}$. A *path type* $\pi$ is a sequence of relations $r_1 - r_2 - \ldots - r_l$, some of which may be the inverses of relations in $\mathcal{R}$ (the inverse of relation $r$ is denoted by $r^-$). A given path $\pi$ *holds* for entities $h$ and $t$ if there exists a set of entities $e_1, e_2, \ldots$ so that all the variables $\{y_{h,r_1,e_1}, y_{e_1,r_2,e_2}, \ldots y_{e_{l-1},r_l,t}\}$ have value 1. We assume a set $\Pi$ of permissible path types is specified (by the RS designers) so that those path types capture sensible connections between entities [18].

Suppose an RS suggests item $e_i$ to user $e_u$ (note that items and users are represented by entities in the assumed auxiliary KG). A reason *for* this recommendation is simply taken to be a path $\pi \in \Pi$ that takes $e_i$ to $e_u$ in the KG. Thus we have an function $f$ that starts with the KG and the path $\pi$, takes inputs $e_i$ and $e_u$, and returns a set of reasons for the recommendation of $e_i$ to $e_u$. While this function can be implemented in several ways, in our implementation (described later) we employed depth-first search in the KG [18].




**(a) Reason for recommending Red Phone.**

**(b) Reason for recommending Green Phone.**

**Figure 1: Examples of reasons for and reasons against in item-based recommendation.**

To illustrate, Figure 1a shows through graphs an example where the recommendation of the Red Phone to a user is explained by the path $\pi_3$ = (bought, has, has$^-$), which goes through entities User, Laptop, Cutting Edge OS and Red Phone.

### 3.2 Reasons Against: What They Are, and How to Generate Them

We now focus on the main technical challenge in this work: how to generate reasons *against* a particular recommendation. To do so, we resort to the literature on practical reasoning in Philosophy, where we find Snedegar's rather comprehensive theory of reasoning [23]. Snedegar presents five schemes by which reasons *against* can be generated by an agent contemplating competitive options:

- **Scheme 1 (S1)** : a reason against an item $A$ is a reason for a competing option;
- **Scheme 2 (S2)** : a reason against an item $A$ is only a reason for NOT $A$ (not for any particular other option);
- **Scheme 3 (S3)** : a reason against an item $A$ is just a reason for the disjunction of the other options (say $B \vee C \vee D$);
- **Scheme 4 (S4)** : a reason against an item $A$ is a reason for each, i.e. all, of the alternatives to it.
- **Scheme 5 (S5)** : a reason against an item $A$ explains (or is part of the explanation as to) why $A$ promotes or respects some objective less well than some other option.[1]

These schemes have been defined by Snedegar at a highly abstract level; we must taken them to a concrete level. We present our implementations in the remainder of this section.

Our implementation of S1 generates a reason against a given item by generating reasons for other options. For instance, take the case where the RS has recommended two phones — Red and Green — as in Figure 1. A reason against the Red Phone then would be that the Green Phone has a “Long Duration Battery”.

Scheme S2 is more delicate: how to define the negation of an item in the context of recommendations? The vague nature of this question led us to skip this scheme.

[1] This scheme requires one to specify a quantitative objective.

Our implementation of S3 goes through all competing options, collecting reasons for them that are not reasons for the option of interest; we then trim the list of reasons against to an arbitrary small number of reasons (e.g. 3). In our running example we can imagine there is a Blue Phone and as reasons against the Red Phone we have that the Green Phone, the Blue Phone or both of them have long duration batteries. In practice S1 and S3 produce identical reasons against.

The implementation of S4 is similar to that of S3 to the extent that S4 takes reasons for all competing options into account (reasons against according to S4 are also reasons against according to S3). An example of reason against the Red Phone using S4 would be that both the Blue Phone and the Green Phone from the example above have adequate battery duration. The stringent nature of this scheme, where the intersection of reasons is required, makes it hard to generate reasons against in practical circumstances.

Scheme S5 depends on a quantitative objective that can be the basis of explanations; this objective is used to determine whether a reason is for or against an option. Consider in our phone example that the user has the objective of long battery life for her phone; with that piece of information, the RS can present the user with the reason against buying the Red Phone because it has a short duration battery. We have implemented S5 by assuming that an objective function is known; however, this is not a realistic assumption and future work should address the elicitation of objectives at running time.

To illustrate the implemented algorithm, suppose an RS recommended $N$ items in an ordered set $\mathcal{I} : \{i_1, i_2, ...i_N\}$ to user $u$. In Schema S1 (and S3) we define as reason against an item $i_r$ the union of reasons for each of its alternatives $\mathcal{I} \backslash \{i_r\}$ that are not reasons for $i_r$ itself. Hence we must iterate over the alternatives, extracting reasons for each one of them $\Phi \leftarrow \Phi \cup \Phi_{u,i}\ \forall i \in \mathcal{I} \backslash \{i_r\}$. Note that at this point we assume that function $f$, as described in Section 3.1, is available. We then remove from $\Phi$ the reasons for our recommendation of interest, if any. The remaining reasons $\Omega = \Phi / \Phi_{u,i_r}$ are the reasons against $i_r$ – as presented in the Algorithm 1.

Regarding the implementation of Schema 4 (S4), we follow a very similar procedure, except that instead of considering the union of reasons for its alternatives, we take the intersection. That is, we just replace the line 15 of the Algorithm 1 so as to take the intersection of sets $\Phi \leftarrow \Phi \cap \Phi_{u,i}\ \forall i \in \mathcal{I} \backslash \{i_r\}$.

To close this section, consider an extended example using Scheme S1. We focus on Scheme 1 due to the fact that it captures most of the content of Scheme S3 as well; as noted already, Scheme 2 does not seem conducive to a concrete implementation, and Scheme 5 requires elicitation of user objectives — finally, as discussed later in connection with our experiments, Scheme 4 does not seem very promising in practice.

**Algorithm 1** Explanation Generation using Scheme S1

1: **procedure** REASONS-FOR($i, u, \Pi, \mathcal{G}$)
2: $\quad\Phi_{u,i} = \{\}$ ▷ Set of reasons for $i$
3: $\quad$**for all** $\pi \in \Pi$ **do**
4: $\quad\quad\phi \leftarrow f(u, i, \pi|\mathcal{G})$ ▷ Function described in Section 3.1
5: $\quad\quad\Phi_{u,i} \leftarrow \Phi_{u,i} \cup \phi$
6: $\quad$**end for**
7: $\quad$**return** $\Phi_{u,i}$
8: **end procedure**
9: **procedure** REASONS-AGAINST-S1($i_r, u, \mathcal{I}, \Pi, \mathcal{G}$)
10: $\quad\Omega_{u,i_r} \leftarrow \{\}$ ▷ Set of reasons against $i_r$
11: $\quad\Phi = \{\}$
12: $\quad\Phi_{u,i_r} \leftarrow$ REASONS-FOR($i_r, u, \Pi, \mathcal{G}$) ▷ Set of reasons for $i_r$
13: $\quad$**for** $i \in \mathcal{I} \backslash \{i_r\}$ **do** ▷ Iterate over $i_r$ alternatives
14: $\quad\quad\Phi_{u,i} \leftarrow$ REASONS-FOR($i, u, \Pi, \mathcal{G}$)
15: $\quad\quad\Phi \leftarrow \Phi \cup \Phi_{u,i}$
16: $\quad$**end for**
17: $\quad\Omega_{u,i_r} \leftarrow \Phi \backslash \Phi_{u,i_r}$
18: $\quad$**return** $\Omega_{u,i_r}$
19: **end procedure**

*Example 3.1.* We have built an RS to suggest University classes called Ganimedes. A student asks for courses by presenting a few topics to Ganimedes; the RS then uses information from syllabuses and an associated knowledge graph to produce recommendations. The knowledge graph, called USPedia, collects information about topics and their relationships; it was automatically harvested from Wikipedia pages [18]. We have defined a number of permissible paths for explanations (Section 3.1). For instance, one of them is (subject, broader$^-$, broader); as this permissible path indicates that a subject is of the same broader category as another topic of interest. That is, subject$(X, Y)$ broader$(Z, Y)$ broader$(Z, W)$ means that Y is a topic of X, Z has the same broader categories of Y and W and, finally, that W is of the same broader category of a topic of X.

We assume that a course is likely to be about a given subject when it deals with topics that are related to that subject. For instance, a student who is a machine learning (ML) enthusiast would be satisfied with a course that is about statistical models even if the course is not focused on ML itself.

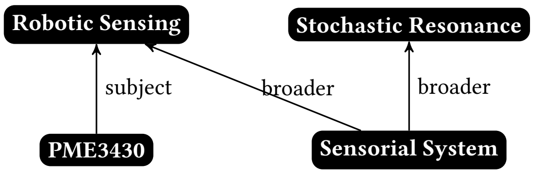
Figure 2 conveys a number of explanations generated by our RS. In this case, the student asked our RS for courses about *Stochastic Resonance*, and was suggested classes with codes PME3430 and PME3479. The RS found two reasons for PME3430 (Fig. 2a and Fig. 2b) and one for PME3479 (Fig. 2c). Note that both recommendations share the *reason for* depicted in Fig. 2b and 2c, thus it cannot be a reason against for none of them. On the other hand, the one in Fig. 2a is a reason for only PME3430; therefore, it is a reason against PME3479. □

## 4 EXPERIMENTS

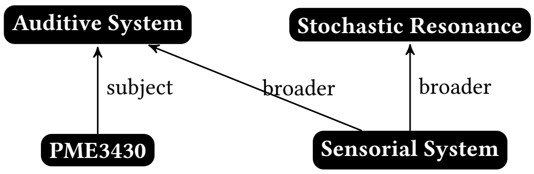
In this section we describe experiments with simulated and real users; we first examine the feasibility of our techniques in Section 4.1 and then we discuss the reaction of human users to our approach in Section 4.2.

### 4.1 Evaluation of feasibility: simulated interactions

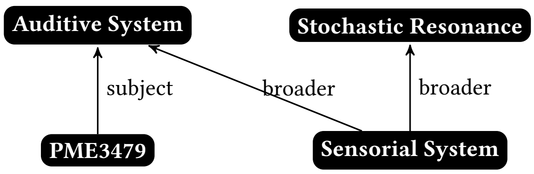
We have first evaluated our proposal from two perspectives: (1) the fraction of recommendations for which we can find at least one explanation (we refer to it as *coverage*) and (2) the average number of reasons we can find to support/attack a given recommendation (we refer to it as *support*) [18, 27]. These metrics offer a glimpse at the workings of our proposal in a real-world scenario from a




(a) Reason for PME3430 and against PME3479.




(b) Reason for PME3430.




(c) Reason for PME3479.

**Figure 2: Examples of S1 with two reasons for and against the recommendation PME3430.**

objective perspective. To carry out our experiments, we trained an RS based on TransE [2] embedding from the USPedia knowledge graph employed in Example 3.1, using the same set-up as in Ref. [18]. We built our simulated interactions by asking for the Top-4 recommendations of randomly sampled 100 cases. Next, for each interaction, we used our proposed method to retrieve both reasons for and against.

Regarding *reasons for*, Table 1 shows that we obtained 79.33% coverage and a support mean of 2.0, similar results to those reported in previous works [18, 27]. As for *reasons against*, we ran our experiments considering Schemas S1 and S4. Both the coverage (85.1%) and support (2.3) obtained for S1 are higher than those from *reasons for*. This result was expected since S1 implementation considers more aggregated reasons for alternatives than it removes from the recommendation being explained.

On the other hand, Scheme S4 could *not* generate a single reason against at all (coverage 0%!). As Scheme S4 requires that a reason against an option must be a reason for all of its alternatives, it imposes a restriction so rigorous that it is in fact unfeasible in practice.

| Explanation Type | Coverage | Support |
|---|---|---|
| **Reason For** | 79.3% | $2.0 \pm 1.0$ |
| **Reason Against (S1)** | 85.1% | $2.3 \pm 1.4$ |
| **Reason Against (S4)** | 0% | - |

**Table 1: Coverage and Support for reasons for and reasons against using Schemas S1 and S4.**

| Metric | Question |
|---|---|
| transparency | The explanation on the right helped me understand why the items were recommended better than the explanation on the left |
| persuasion | Based on the explanation on the right, I was more prone to follow the recommendation than based on the explanation on the left |
| engagement | The explanation on the right helped me learn more about the recommended items than the explanation on the left |
| trust | The explanation on the right contributed more to increase my confidence in the recommendations than the explanation on the left |
| effectiveness | The explanation on the right made me more confidence about making the best choice than the explanation on the left |

**Table 2: The five explanation metrics that subjects had to take into account in the experiment.**

## 4.2 Evaluation with Human Subjects

One could expect the fact that an RS can be built with reasons for/against does not mean that human subjects would be satisfied with it; to determine whether indeed our approach is a valuable one, we carried out an experiment to address the following questions:
1) Do reasons for/against have value for users?
2) Do reasons against reduce an RS persuasion?
3) Do users perceive a conflict of interest in their interaction with an RS?
4) Do reasons for/against influence user choices?

Our experiment took 31 subjects, all of which are undergraduate students, and asked them to evaluate two RS implementations, one displaying only reasons for recommendations, and the other displaying reasons for and against them. Subjects were presented with an e-commerce mock-up where they received recommendations concerning smartphones. Each subject first received a recommendation and one reason for, and was asked to select an item; then the subject received a recommendation with one reason for and one reason against, and was again asked to select an item. Note that we avoided presenting too many reasons at once. Figure 3 depicts the information presented.

Each subject then evaluated the two RSs individually using five *explanation metrics* [25] that are presented in Table 2. Each subject ranked each RS with respect to each explanation metric using a survey-based Likert psychometric scale [9] from 1 to 5 (standing for "Strongly disagree", 2 "Disagree", 3 "Neither agree nor disagree", 4 "Agree", and 5 "Strongly agree"). This scale was used to reduce central tendency and social desirability biases where subjects do not want to be identified with extreme positions. Finally, each subject could write a short free text with thoughts about the RSs.

Figure 4a shows the percentage of responses given by subjects. Responses, notably for *engagement*, *trust* and *effectiveness*, are concentrated around scores 4 and 5. This result indicates that users mostly agree that showing reasons against a recommendation adds

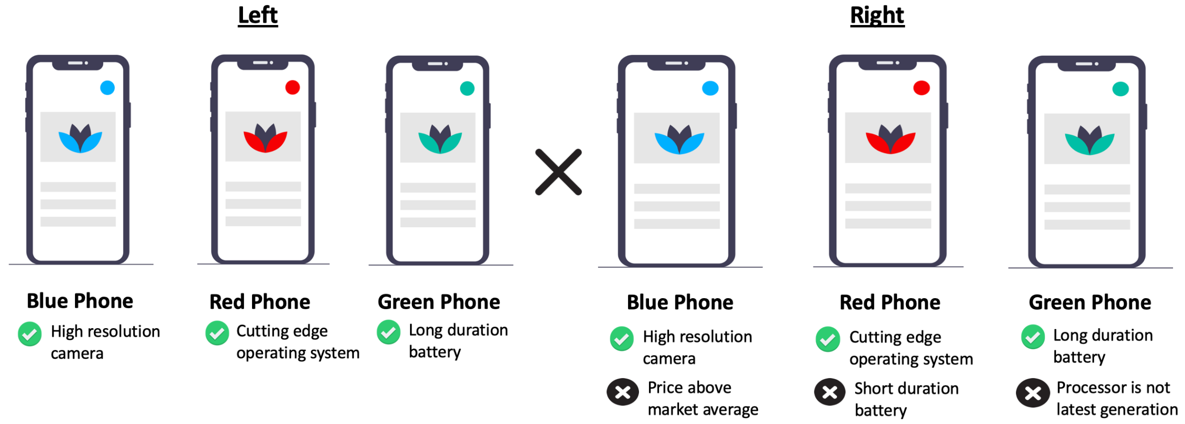



**Figure 3: Experiment: just one reason for (left); one reason for and one reason against (right).**

value with respect to trust, engagement and effectiveness of RS. Figures 4a and 4b show that there was a divergence amongst users about whether the proposed explanation paradigm increases transparency. As our method is model-agnostic (it makes no assumptions about the RS internal behavior), the explanations were unable to shed light on how items were actually recommended. As the transparency score peaked around 3, this does not mean reasons for/against were adverse to transparency; it means that they were as good as just reasons for.

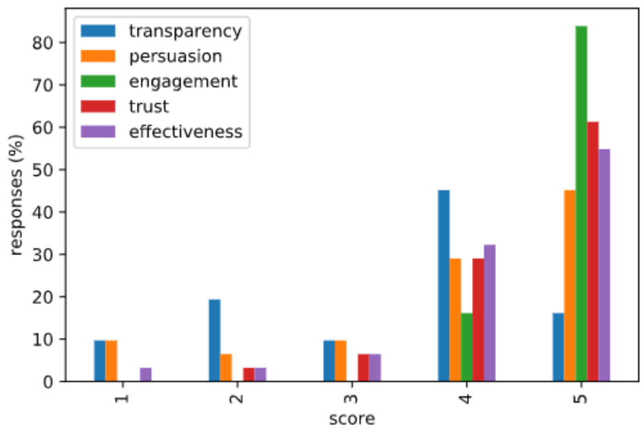
We expected a possible drawback of our proposal would be a reduction in persuasion (as reasons against might make the users less likely to follow recommendations). Figure 4b shows that the down whisker is longer for persuasion than it is for trust, engagement and effectiveness. However, note that the boxplot for persuasion is skewed up; thus most users felt more convinced when reasons against were present. By doing a further analysis of textual comments, we found out that persuasion increases are produced by higher trust in the RS. Consider two comments:

> 1) *I always think that recommendations that bring positive and negative aspects are fairer, and could influence me more into buying the product, once I feel I am not being misled.*
> 2) *As the first example [the first RS] shows only strong points for each product, it leads the user to have a certain mistrust about the suggestions.*

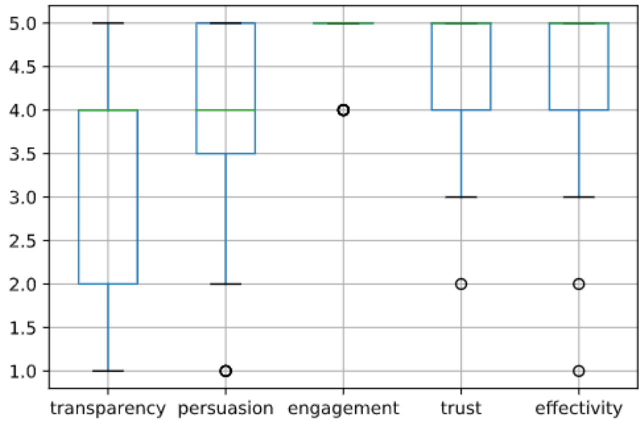
Comments also indicate that many users expect the RSs to try to lead them into a decision, sensing a conflict of interest in the process. Consider the following comment:

> 3) *Differently from marketing which always idealize the product, this one seems to show the reality about it, thus I feel I understand the recommended product in its real form.*

These comments corroborate our hypothesis that, indeed, reasons against have a significant positive impact on the user decision-making process. As a matter of fact, a full 45% of our test subjects changed their choices after we presented reasons against.




**(a) Visual representation for explanation metrics average scores.**




**(b) Boxplots for the explanation metrics.**

**Figure 4: Results from the experiment with human subjects.**

# 5 CONCLUSION

In this paper we have proposed a novel feature for RSs, whose goal is to enhance trust by acting responsibly; namely, we investigated the generation of reasons *for* and *against* recommendations. By displaying such reasons, an RS not only helps the user to reach the most rewarding decision, but the RS acts on its own interest in building trust.

We have developed ways to generate reasons for/against using an auxiliary KG by adapting Snedegar's theory of practical reasoning. Our implementation demonstrates that additional calculations needed to generate such reasons do not affect overall performance. By implementing Snedegar's theory we have found difficulties with some of his schemes for reasons against; we suggest that his Scheme 1 is the most appropriate in practice at the moment. Moreover, our experiment with human subjects demonstrated that reasons against can significantly increase trust, engagement, and even persuasion. Overall we demonstrated that adding reasons against items does improve RSs.

Future work should investigate how much information should be given to users when presenting reasons for/against. It would also be useful to explore mental models of the user so as to extract quantitative objectives to use in Snedegar's Scheme S5. Moreover, it would be important to evaluate our proposals at scale.

# 6 ACKNOWLEDGMENTS

This work was carried out with the support of Itaú Unibanco S.A.; the second author has been supported by the Itaú Scholarship Program (PBI), linked to the Data Science Center (C2D) of the Escola Politécnica da Universidade de São Paulo. The third author has been partially supported by the Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq), grant 312180/2018-7. This work has been partially supported by the Fundação de Amparo à Pesquisa do Estado de São Paulo (FAPESP), grant 2019/07665-4. We acknowledge support by CAPES finance code 001. We are also grateful to the Center for Inovation at Universidade de São Paulo (InovaUSP) for hosting our lab.

# REFERENCES

[1] M. Alshammari, O. Nasraoui, and S. Sanders. 2019. Mining Semantic Knowledge Graphs to Add Explainability to Black Box Recommender Systems. *IEEE Access* 7 (2019), 110563–110579. https://doi.org/10.1109/ACCESS.2019.2934633

[2] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Durán, Jason Weston, and Oksana Yakhnenko. 2013. Translating Embeddings for Modeling Multi-Relational Data. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2* (Lake Tahoe, Nevada) *(NIPSâĂŹ13)*. Curran Associates Inc., Red Hook, NY, USA, 2787âĂŞ2795.

[3] Finale Doshi-Velez and Been Kim. 2017. Towards A Rigorous Science of Interpretable Machine Learning. http://arxiv.org/abs/1702.08608 cite arxiv:1702.08608.

[4] Evan Estola. 2016. When Recommendation Systems Go Bad. In *Proceedings of the 10th ACM Conference on Recommender Systems* (Boston, Massachusetts, USA) *(RecSys âĂŹ16)*. Association for Computing Machinery, New York, NY, USA, 367. https://doi.org/10.1145/2959100.2959117

[5] Ruining He, Wang-Cheng Kang, and Julian McAuley. 2017. Translation-Based Recommendation. In *Proceedings of the Eleventh ACM Conference on Recommender Systems* (Como, Italy) *(RecSys âĂŹ17)*. Association for Computing Machinery, New York, NY, USA, 161âĂŞ169. https://doi.org/10.1145/3109859.3109882

[6] Veronika Henk, Sahar Vahdati, Mojataba Nayyeri, Mehdi Ali, Hamed Shariat Yazdi, and Jens Lehmann. 2018. Metaresearch Recommendations using Knowledge Graph Embeddings.

[7] Xiao Huang, Jingyuan Zhang, Dingcheng Li, and Ping Li. 2019. Knowledge Graph Embedding Based Question Answering. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining* (Melbourne VIC, Australia) *(WSDM âĂŹ19)*. Association for Computing Machinery, New York, NY, USA, 105âĂŞ113. https://doi.org/10.1145/3289600.3290956

[8] Y. Koren, R. Bell, and C. Volinsky. 2009. Matrix Factorization Techniques for Recommender Systems. *Computer* 42, 8 (2009), 30–37.

[9] Rensis Likert. 1932. A Technique for the Measurement of Attitudes. *Archives of Psychology* 140 (1932), 1–55.

[10] Hao Ma, Dengyong Zhou, Chao Liu, Michael R. Lyu, and Irwin King. 2011. Recommender Systems with Social Regularization. In *Proceedings of the Fourth ACM International Conference on Web Search and Data Mining* (Hong Kong, China) *(WSDM âĂŹ11)*. Association for Computing Machinery, New York, NY, USA, 287âĂŞ296. https://doi.org/10.1145/1935826.1935877

[11] Paolo Massa and Paolo Avesani. 2007. Trust-Aware Recommender Systems. In *Proceedings of the 2007 ACM Conference on Recommender Systems* (Minneapolis, MN, USA) *(RecSys âĂŹ07)*. Association for Computing Machinery, New York, NY, USA, 17âĂŞ24. https://doi.org/10.1145/1297231.1297235

[12] Silvia Milano, Mariarosaria Taddeo, and Luciano Floridi. 2020. Recommender systems and their ethical challenges. *AI & SOCIETY* 35 (2020). https://doi.org/10.1007/s00146-020-00950-y

[13] Tim Miller. 2019. Explanation in artificial intelligence: Insights from the social sciences. *Artificial Intelligence* 267 (2019), 1 – 38. https://doi.org/10.1016/j.artint.2018.07.007

[14] Christoph Molnar. 2019. *Interpretable Machine Learning*. [Online]. https://christophm.github.io/interpretable-ml-book/.

[15] Cataldo Musto, Fedelucio Narducci, Pasquale Lops, Marco de Gemmis, and Giovanni Semeraro. 2019. Linked open data-based explanations for transparent recommender systems. *International Journal of Human-Computer Studies* 121 (2019), 93 – 107. https://doi.org/10.1016/j.ijhcs.2018.03.003 Advances in Computer-Human Interaction for Recommender Systems.

[16] Maximilian Nickel, Kevin Murphy, Volker Tresp, and Evgeniy Gabrilovich. 2015. A Review of Relational Machine Learning for Knowledge Graphs. https://doi.org/10.1109/JPROC.2015.2483592 cite arxiv:1503.00759Comment: To appear in Proceedings of the IEEE.

[17] John OâĂŹDonovan and Barry Smyth. 2005. Trust in Recommender Systems. In *Proceedings of the 10th International Conference on Intelligent User Interfaces* (San Diego, California, USA) *(IUI âĂŹ05)*. Association for Computing Machinery, New York, NY, USA, 167âĂŞ174. https://doi.org/10.1145/1040830.1040870

[18] G. P. Polleti, H. N. Munhoz, and F. G. Cozman. 2020. Explanations within Conversational Recommendation Systems: Improving Coverage through Knowledge Graph Embedding. In *2020 AAAI Workshop on Interactive and Conversational Recommendation System*. AAAI Press, New York City, New York, USA.

[19] Marco T. Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. Model-Agnostic Interpretability of Machine Learning. arXiv:1606.05386.pdf http://arxiv.org/abs/1606.05386.pdf

[20] Francesco Ricci, Lior Rokach, Bracha Shapira, and Paul B. Kantor. 2010. *Recommender Systems Handbook* (1st ed.). Springer-Verlag, Berlin, Heidelberg.

[21] Ruslan Salakhutdinov and Andriy Mnih. 2007. Probabilistic Matrix Factorization. In *Proceedings of the 20th International Conference on Neural Information Processing Systems* (Vancouver, British Columbia, Canada) *(NIPSâĂŹ07)*. Curran Associates Inc., Red Hook, NY, USA, 1257âĂŞ1264.

[22] Rashmi Sinha and Kirsten Swearingen. 2002. The Role of Transparency in Recommender Systems. In *CHI '02 Extended Abstracts on Human Factors in Computing Systems* (Minneapolis, Minnesota, USA) *(CHI EA '02)*. ACM, New York, NY, USA, 830–831. https://doi.org/10.1145/506443.506619

[23] Justin Snedegar. 2018. Reasons for and reasons against. *Philosophical Studies* 175, 3 (2018), 725–743.

[24] Maartje ter Hoeve, Anne Schuth, Daan Odijk, and Maarten de Rijke. 2018. Faithfully Explaining Rankings in a News Recommender System. *CoRR* abs/1805.05447 (2018). arXiv:1805.05447 http://arxiv.org/abs/1805.05447

[25] Nava Tintarev and Judith Masthoff. 2007. A Survey of Explanations in Recommender Systems. In *Proceedings of the 2007 IEEE 23rd International Conference on Data Engineering Workshop (ICDEW '07)*. IEEE Computer Society, Washington, DC, USA, 801–810. https://doi.org/10.1109/ICDEW.2007.4401070

[26] W3. 2019. RDF 1.1 Concepts and Abstract Syntax. https://www.w3.org/TR/2014/REC-rdf11-concepts-20140225/

[27] Wen Zhang, Bibek Paudel, Wei Zhang, Abraham Bernstein, and Huajun Chen. 2019. Interaction Embeddings for Prediction and Explanation in Knowledge Graphs. *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining* (Jan 2019). https://doi.org/10.1145/3289600.3291014

[28] Yongfeng Zhang and Xu Chen. 2018. Explainable Recommendation: A Survey and New Perspectives. *CoRR* abs/1804.11192 (2018), 1–101. arXiv:1804.11192 http://arxiv.org/abs/1804.11192

# ANNEX B – RELATED PAPER - UMUAI 2023

The following paper is under review by the UMUAI Special Issue on Conversational Recommender Systems: Theory, Models, Evaluations, and Trends.

We previously sent an expanded abstract and received approval for submitting the journal paper.

# Generating Explanations for Knowledge-Aware Conversational Recommendation Systems

Gustavo P. Polleti†, Douglas L. de Souza† and Fabio G. Cozman

Escola Politécnica, Universidade de São Paulo, Av. Prof. Luciano Gualberto, São Paulo, 05508-010, SP, Brazil.

*Corresponding author(s). E-mail(s): gustavo.polleti@usp.br;
Contributing authors: douglas.luan.souza@usp.br; fgcozman@usp.br;
†These authors contributed equally to this work.

**Abstract**

Conversational recommendation systems can greatly benefit from techniques that explain the reasons behind their actions. We propose techniques that generate explanations by resorting to an auxiliary knowledge graph and an associated knowledge embedding. By exploiting the embedding plausibility score in depth first search, we present a method that effectively generates reasons for a recommendation. We then propose a host of techniques to generate balanced reasons both for and against a recommendation, so as to enhance user trust in the conversational recommendation system. To do so, we develop a concrete implementation of Snedegar's theory of reasons for/against. Experiments at functional, human, and application levels demonstrate that our proposals do improve the interpretability of conversational recommendations systems with controlled computational cost.

**Keywords:** Conversational Recommendation Systems, Explainable AI, Interpretability, Knowledge Graphs

## 1 Introduction

Regardless of how accurate a recommendation system may be, users can always reject recommendations they do not trust. Not surprisingly, the interpretability of recommended options has received significant attention in the literature on recommendation systems [1–3]. Detailed studies suggest that users prefer to follow recommendations that are properly escorted by some sort of explanation [4].

In this paper we introduce a number of novel, model-agnostic, explanation generation techniques for knowledge-aware conversational recommendation systems.

The fact that we deal with *conversational* recommendation systems (CRSs) raises a number of specific challenges [5]. While existing explanation generation methods are designed to work offline and may take up to minutes to generate an explanation [6, 7], conversational recommendation systems operate online and must therefore be responsive; if an explanation is to be provided, then it must be produced quickly. No user will wait a minute to grasp the reasons why a particular product has been suggested.

Explanation generation methods that meet tight time constraints typically employ large-scale knowledge graphs to display connections between chosen and recommended items [8–10]. However, the effectiveness of such methods is limited because knowledge graphs themselves are often incomplete [11]; many recommendations may remain unexplained due to missing links [12, 13]. This is particularly damaging for conversational recommendation systems because users can be easily disappointed if the system fails to respond when asked for an explanation. Our proposed techniques, which benefit from knowledge embeddings [11, 14], deal with time constraints and knowledge graph incompleteness simultaneously.

Although existing explanation methods do reach some degree of interpretability [1–3], such methods typically provide only positive aspects of a recommendation [15, 16]. This does not offer a balanced perspective to the user, ultimately squandering overall trust [17]. Such methods might be compared to a salesperson who always proposes products with complimentary words, as opposed to the salesperson who frankly discusses the advantages and disadvantages of the products; a perceptive customer will gradually favor a salesperson who chooses sincerity over persuasion. Similarly, a proper explanation method should fully inform the human end-users so they can make an informed decision when choosing to either follow or to reject a recommendation.

We use Snedegar's theory of reasons [18], a philosophical theory of practical reasoning, to develop explanation generation methods that provide explanations with both reasons for and against the system's recommendations. In short, our methods start from existing procedures that generate reasons *for* by processing and analyzing paths in knowledge graphs [10, 19]. We then modify such procedures so as to detect paths (or their absence) that count as reasons *against*. Snedegar's theory relies on five schemes of "reasons against"; we examine their computational implementation in conversational recommendation systems and identify the most promising strategies.

We have evaluated our approach through user studies (to examine overall trust in balanced explanations) and through simulated interactions with a real-world data-set (to evaluate performance in several dimensions). We show that our proposed methods are feasible and able to meet strict time constrains for conversational recommendation systems. The user studies show evidence that our techniques have a beneficial impact on the user perspective.

The paper is organized as follows. Section 2 presents some basic notions on general and conversational recommendation systems, interpretability, knowledge graphs and embeddings, mostly to fix terminology. In Section 3 and Section 4 we propose novel techniques and describe them in detail. We then present our experiments and analysis in Sections 5 and 6, and offer concluding remarks in the last section.

# 2 Background

This section summarizes a few relevant issues related to conversational recommendation systems and to interpretability (Section 2.1), as well as basic notions related to knowledge graphs and knowledge embeddings (Section 2.2). It then reviews literature that is directly tied to the topic of this paper (Section 2.3).

## 2.1 Recommendation Systems and Interpretability

Recommendation systems support decision-making processes in scenarios with a large variety of options [20]. Primitive recommendation systems can be traced back to rankings of best movies or lists of top ten artists that can still be seen in magazines and newspapers to date. Those rudimentary tools, however, do not take into account user-specific intentions. Recommendation systems, on the other hand, take user-specific interests to be a central guide in finding recommendations.

There are many ways to implement a recommendation system, depending on how the system handles similarities among items and users. Implementations have been classified into six different groups: Content-based, Collaborative filtering, Demographic, Knowledge-based, Community-based and Hybrid [21]. In practice, successful strategies often take a hybrid approach [20].

One way to build content-based systems is to map semantically rich features into numerical vectors, so as to operate with numerical vectors that capture the underlying semantics. Such *embeddings* are expected to map similar items to nearby vectors; thus one can select items that are similar to any given item.

State-of-the-art recommendation systems take into account substantial information, to the point that some systems seem to "know more about users preferences than the users themselves" [22]. However, this improvement in accuracy has produced opaque models, often based on embeddings, that harm overall interpretability [23]. Here we define interpretability, rather concisely, as the degree to which a human can understand the cause of a decision [24]. Note that a device may be transparent in that the user can access all elements of its operation, yet its output may have low interpretability. When interpretability is low, one possible strategy is to generate explanations for the decisions [25–27].

An explanation can be perceived as the answer to a "why" question [24]. For example, popular methods from the literature typically answer the question "why did the model yield a certain output given its input?" To answer such a question, explanations list the features or input values most correlated with the model output [28]. These methods explain outputs on the assumption that features themselves are interpretable, a weak assumption in the context of embeddings and recommendation systems. When applied to embeddings, explanations might be, for instance, that "dimensions 158 and 254 are significant for the recommendation" — a sentence that is hard to interpret as such an explanation convey little meaning. We obviously hope that every recommendation can be properly explained; in practice, a system may fail to provide explanations in some cases. The fraction of recommendations that are explained is the *coverage* of the explanatory process.

Furthermore, instead of understanding why a given item was recommended, users may be more interested in why they should follow a given recommendation and not its alternative options. Hence the goal of explanation generation in recommendation systems may differ from the objectives in most of the explainability literature.

The act of explaining requires a social interaction between at least two agents: the explainer and the explainee. The explanation model proposed by Miller argues that the explanation is a compound process with two steps: the cognitive and the social. While the first describes the process of identifying the causes of why a given decision was made, the second is the process of conveying or communicating such reasons to the explainee. Figure 1 offers a visualization of this process, that starts with the opaque black-box model, for which we find the explanations comprising the cognitive and social phases; then the explanation is given to the inquirer, the explainee.

## 2.2 Knowledge Graphs and Knowledge Embeddings

An emergent class of state-of-the-art content-based techniques, knowledge-aware ones, typically benefit from knowledge graphs to model item contents [14, 29]. Such structures provide enough flexibility to describe items through multiple sources and formats (e.g. tabular, image, text, etc) so that the recommendation content can be modeled more finely than traditional user-item interaction matrices. In this work we analyze graph-like approaches from the unified perspective of knowledge graphs and their related embedding models, often referred to as *knowledge embeddings*.

We loosely follow RDF notation [30], focusing on data-sets where a triple representing a fact is denoted by $\langle h, r, t\rangle$ where $h$, $r$ and $t$ are, respectively, the *subject (head)*, *predicate (relation)* and *object (tail)*. A knowledge graph $\mathcal{KG}$ is defined by triples containing entities in set $\mathcal{E} = \{e_1, \ldots, e_{N_e}\}$ and relations in set $\mathcal{R} = \{r_1, \ldots, r_{N_r}\}$. The presence/absence of a triple $x_{h,r,t} = \langle h, r, t\rangle$ is indicated by a random variable $y_{h,r,t} \in \{0, 1\}$. For instance, the information that "Exoplanets is a topic of Astronomy" can be described as the triple $\langle \mathsf{Exoplanets}, \mathsf{topic\_of}, \mathsf{Astronomy}\rangle$, in which both Exoplanets and Astronomy are entities, and topic_of is the relation connecting them. A knowledge graph consists of a collection of triples where the head and the tail represent nodes and the relation is a directed edge connecting head to tail.

Large-scale knowledge graphs often have limited applicability due to their severe incompleteness. For instance, the place of birth attribute was missing for 71% of all people in Freebase in 2012 [11]. While existing triples encode known true or positive relationships between entities, there is no explicit representation for false relationships in a knowledge graph. For instance, the statement: "JaneDoe has a son called BobDoe" can be encoded by the triple $\langle \mathsf{JaneDoe}, \mathsf{son}, \mathsf{BobDoe}\rangle$. However the exact opposite, i.e. "JaneDoe does not have a son called BobDoe" is usually encoded by the absence of the triple. Such semantics does not distinguish between false and unknown facts in a knowledge graph; hence the incompletness represents an even more severe limitation.

Several approaches have been developed to address the task of *Knowledge Base Completion* (KBC), i.e. to find true or false facts among the unknown triples in a knowledge graph [11]. Approaches based on knowledge embeddings now display

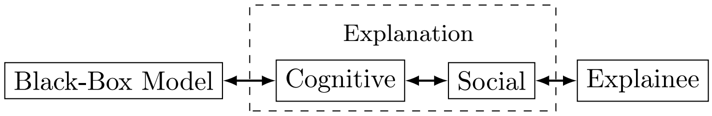



**Figure 1** Visual representation of the explanation processes.

state-of-the-art performance in knowledge base completion [31]. Knowledge embeddings operate by learning latent features from observed data and using them to infer missing facts in the original knowledge graph. Typically, a model based on an embedding defines a particular scoring function $f_r(h, t \mid \Theta)$ to measure the plausibility of fact $\langle h, r, t\rangle$, where $\Theta$ is a set of parameters. The greater the plausibility score for a given fact, the more likely it is to hold. For instance, one can infer the triple $\langle \mathsf{Exoplanets}, \mathsf{topic\_of}, \mathsf{Astronomy}\rangle$ by evaluating the plausibility score $f_{\mathsf{topic\_of}}(\mathsf{Exoplanets}, \mathsf{Astronomy} \mid \Theta)$. Even though the plausibility function depends on the embedding model, its intuition is similar to a measure of distance between knowledge graph entities represented in an embedding space. The closer the entities embeddings are in this learned space, the more plausible their relationship is to hold. To illustrate, one simple plausibility function is the Euclidean norm of the distance between the tail and the head entity embeddings after a linear translation, i.e. $- \parallel \mathbf{h} + \mathbf{r} - \mathbf{t} \parallel_{\frac{1}{2}}$; this is exactly the TransE model [32].

Despite being originally proposed for knowledge base completion, knowledge embeddings are also used to produce recommendations [14, 33]. Proposals in the literature typically employ the plausibility scoring function to rank entities and to return those with the highest plausibility as recommendations. Even though these approaches are accurate, they are not interpretable as they operate in the latent space of embeddings, and they do not attempt to generate explanations for/with embeddings (as we do).

## 2.3 Finding Explanations in a Graph

There has been significant effort to endow conversational recommendation systems with explanatory facilities. In this section we review the relevant literature about this topic. We first discuss methods that build explanations connected with knowledge embeddings, and then we will examine approaches that build explanations out of varied sources.

There are several techniques that try to interpret the content of specific knowledge embeddings, such as SimplE [34], ITransF [35] and CrossE [36]. SimplE looks at each dimension of an entity embedding as a feature, and takes the corresponding relation representation as a measure of that feature's importance to the relation. Even though this approach reaches a certain degree of transparency, it does not really interpret the embeddings. Similarly to SimplE, ITransF also deals with interpretability at the level of individual mappings between entities and relations to vectors. ITransF proposes a version of the attention mechanism, usually employed to define the transformer architecture [37], to find shared concepts among relations. For instance, relations nominated_for and honored_for are both connected to similar concepts even though they are distinct. The attention mechanism of ITransF identifies concepts, but does not really interpret them. Insights generated by SimplE and ITransf provide some degree of interpretability, mostly understandable by experts, but those techniques focus on internal structure encapsulated through numeric quantities.

On the other hand, CrossE exploits a particular type of interaction between relations and entities, referred to as *crossover interactions*, to explain embedding predictions through symbolic means. As such, CrossE is clearly related to proposals we advance later. To illustrate the behavior of CrossE, suppose one has to explain the triple $\langle \mathsf{John}, \mathsf{father\_of}, \mathsf{Bob} \rangle$. An explanation that supports this triple could be the path $\xrightarrow{\mathsf{has_wife}} \xrightarrow{\mathsf{has_child}}$ connecting head and tail entities. The output of CrossE is easily interpretable and faithful to the underlying knowledge graph, but CrossE cannot explain all triples as it is affected by incompleteness of the knowledge graph; another drawback of CrossE is that it is not affected by negative instances (absence of triples). Furthermore, CrossE restricts its explanations to crossover interactions, while other proposals in the literature suggest that more expressive types of graph features can produce better explanations [12, 38].

Techniques discussed in the previous paragraphs are model-specific; we now examine a few model-agnostic approaches in the literature.

Gusmão et al. [12] introduced procedures, collectively referred as "eXplainable Knowledge Embeddings" (XKE), that use knowledge embeddings to generate predictions from a given input and then produce explanations as paths extracted using the Subgraph Feature Extraction (SFE) algorithm [38]. These paths, together with the output produced by the knowledge embedding, are used to train an interpretable machine learning model that serves as a surrogate for the complex knowledge embedding to be explained. However, the incompleteness of knowledge graphs causes XKE to miss explanations for some recommendations (that is, XKE has low explanation coverage). In variants of XKE, missing links are "filled in" by the surrogate model itself (a logistic regressor in reported implementations).

The various relevant existing approaches are summarized in Table 1. In the next sections we offer improvements over previous techniques so as to close a few research gaps we have identified in our practical evaluation of systems. We should also mention the work of Mauro et al. [39], which examines strategies to communicate explanations to users; our experimental study is similar to theirs.

## 3 High Coverage Explanations through Embeddings

In this section we describe the sort of Conversational Recommendation System (CRS) we deal with in the present work, and propose techniques to improve coverage through knowledge embeddings. This sets the stage for our main contributions, described in Section 4.

We suppose the CRS receives a query from the user; the query asks for a topic of interest that is represented as an entity $e_h$ in an auxiliary knowledge graph. The CRS must then find items (entities) that are linked to the entity $e_h$ though some relation $r$ that represents the affinity between the user preference and the items. That is, the system must find triples following the pattern $\langle e_h, r, e_t \rangle$ for possibly many tail entities $e_t$. To do so, the CRS must predict links connecting various entities; we

**Table 1** Comparative summary of related works. Note: we use X-DFS-KE to refer to the "Explanations based on DFS over Knowledge Embeddings" proposed by [40]; we use X-DFS-FA to refer to the "Explanations For/Against by DFS" [41].

| Method | Lay User | Agnostic | Faithful | RT | High Cov. | Resp. |
|---|---|---|---|---|---|---|
| SimplE [34] | No | No | - | No | Yes | No |
| ITransF [35] | No | No | - | No | Yes | No |
| CrossE [36] | Yes | No | Yes | Yes | No | No |
| XKE [12] | Yes | Yes | No | Yes | No | No |
| LISTEN [6] | Yes | Yes | Yes | No | Yes | No |
| ExpLOD [42] | Yes | Yes | No | Yes | No | No |
| ASEMF_UIB [19] | Yes | Yes | No | Yes | No | No |
| X-DFS-KE [40] | Yes | Yes | Yes | Yes | Yes | No |
| XFA-DFS [41] | Yes | Yes | Yes | Yes | – | Yes |

suppose that link prediction operates using a selected knowledge embedding. That is, the CRS returns the top $N$ ranked entities, where $N$ is an hyper-parameter fixed for the CRS, and the ranking follows the plausibility score $f_r(e_h, e_t \mid \Theta)$ specified through the embedding; here the score given by the embedding is a predicted measure of how strong the item entities relate to the user preference. Note that parameters $\Theta$ are tied to the embedding and are learned from previously processed training data.

To illustrate, take a system that recommends classes to students based on a given topic of interest. Suppose a student sends the query "recommend me something about Astronomy to which the CRS might produce the output "I recommend you Exoplanets101". The CRS presumably identified Astronomy as the topic (item) of interest, and found the possible classes to recommend by accessing a previously computed list $\mathbb{T}$ containing the plausibility of a link to each entity in its auxiliary knowledge graph through the relation subject. Possibly the CRS had several entities at its disposal in descending order of plausibility, from Exoplanets101 to say Mechanics101, Mechanics202, and so on. In our example, we have

$$\mathbb{T} = \begin{bmatrix} f_{\mathsf{subject}}(\mathsf{Astronomy}, \mathsf{Exoplanets101}|\Theta) \\ \vdots \\ f_{\mathsf{subject}}(\mathsf{Astronomy}, \mathsf{Mechanics202}|\Theta) \end{bmatrix}.$$

Note that this conceptual model of the CRS is agnostic to the specific embedding and plausibility function, as it only assumes the existence of a plausibility function. In short, the CRS recommends the entities that best fill in the query $\langle e_h, r, ?\rangle$, where $r$ is a relation modelling how tail entities meet user preferences $e_h$.

In this context, we assume that explanations should assist users in their decision-making processes, so as to let them choose the best possible options in harmony with *their own* opinions, even if their best decision is not to follow recommendations at all. In this section we focus on techniques that generate *reasons for* selecting a particular recommended item; in the next section we expand the discussion to include *reasons against* items.

We introduce a novel method that produces explanations in a feasible time, fashioned for online and interactive systems as required by CRSs, while resorting to knowledge graphs that may be incomplete. We should emphasize that speed is of the essence: in a CRS, the user must not wait too much. We have tried solutions based on surrogate methods and topic models,[1] only to find that they were too slow. As we demonstrate in our experiments (Section 5), the direct strategy we present here is indeed effective in practice.

An *explanation* is here a path of length at most $L$, where $L$ is an hyper-parameter, starting and ending at selected entities $e_h$ and $e_t$. For instance, we may find the explanation for an example where the user wishes to learn about Astronomy and receives Exoplanets101 by examining the following part of the knowledge graph:

$$\mathsf{Astronomy} \xrightarrow{\mathsf{subject}} \mathsf{Exoplanets} \xrightarrow{\mathsf{subject}} \mathsf{Exoplanets101}.$$

We refer to such a graph fragment as a *reason for* a recommendation.

We build explanations by first selecting a set of paths $\Pi_L$ of length at most $L$; for instance, we may focus on paths that have sequences of links labeled with relation subject, or perhaps by either subject or topic_of. We take the construction of $\Pi_L$ to be domain-dependent; of course one may use all possible paths of length at most $L$, but this may be unfeasible. There is a trade-off, of course: if one selects too few paths, good explanations may be unavailable. We find it reasonable to assume that a professional building a CRS will be able to impose sensible restrictions on possible paths (or perhaps consult a domain expert that will do so). For large knowledge graphs, automated methods such as graph feature selection algorithms [38] may be needed.

To generate an explanation, we go through every path $\pi \in \Pi_L$ starting from $e_h$, using a depth-first search (DFS). If the search reaches $e_t$, the path from $e_h$ to $e_t$ is a possible explanation. Note that the search-tree height is at most $L$; that is why we use DFS instead of, for example, breadth-first search.

The drawback of a DFS that only uses links in fact present in the knowledge graph is *low coverage*, by which we mean that many pairs of entities may not be connected by any path. Ideally, we would like to have a hundred percent coverage, covering every recommendation with an explanation. In preliminary testing, we have observed that the number of paths generated in usual knowledge graphs is rather low, given that such graphs are sparse. We need to improve coverage by moving beyond the edges that are in the knowledge graph; we do so by running the search in an enlarged (weighted) graph.

That is, we wish to run the search in the space containing completions of the knowledge graph as established by the knowledge embedding. However, the embedding is a continuous mapping, not a graph. How do we conduct DFS on it? Our solution is to build a weighted graph $\hat{G}$ from the knowledge embedding. We build $\hat{G}$ by adding links connecting all entities, and weighing those links by their plausibility score as long as the score is higher than a threshold $\delta_r$ (note that there is a threshold per relation). This obviously assumes that the higher the plausibility, the more we should expect the

[1] Some of those solutions have been discussed in previous workshop papers [7, 40, 41]; those publications also mention some of the ideas that we explore, with substantial improvements, in the present paper.

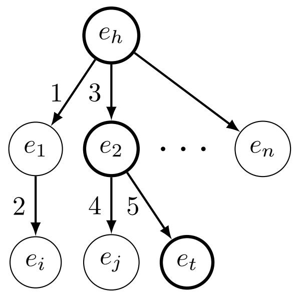



**Figure 2** Depth-First Search toy example.

link to be present — an assumption that is aligned with the behavior of embeddings. Note that we do not have to build a graph in memory; we simply run DFS with the possible moves encoded by the embedding. That is, we prioritize edges with a high plausibility in the DFS. The result is that paths resemble the ones produced by the Path Ranking Algorithm for knowledge graph completion [43].

To summarize, an explanation is a sequence of relations $\pi = \{r_1, r_2, \ldots, r_L\}$, that is built only if we can find triples from the selected $e_h$ to $e_t$ going through the relations, such that each triple $\langle e_i, r_k, e_j \rangle$ either is in the original knowledge base, or displays a plausibility score such that $f_{r_k}(e_i, e_j|\Theta) > \delta_{r_k}$. The DFS starts from $e_h$, moving downward, according to the plausibility scores, to another entity. Even though the search must stop as the number of possible paths is finite, it may be useful to abort the process after some practical time limit (we do so in our implementation).

To illustrate this procedure, consider the example of a DFS in Figure 2. The nodes are sorted from the highest plausibility score on the left to the lowest on the right. The numbers on the edges show the order each node is visited. Please also consider the toy example in Figure 3; it indicates that Exoplanets101 is recommended because it is about Exoplanets, which is a topic of Astronomy. The graph also contains the explanation for recommending Aeronautics102 because it teaches Rocket Science that is also a topic of Astronomy. Note that the student wishes to learn about Astronomy in this case.

# 4 Responsible Explanations through Reasons-Theory

We have developed, in the last section, a model-agnostic method that generates fast and faithful explanations based on an auxiliary knowledge graph and an associated knowledge embedding, while keeping high coverage (that is, a high percentage of recommendations are duly explained). We have thus addressed the *cognitive* phase of the explanatory process, in the sense of Miller [24], as depicted in Figure 1. However, explanations so far do not shed any light on possible downsides of recommended items: Users may feel that the explanations are given solely to persuade them to agree with the recommendation system, and may lose their trust in the system.

We now move to our main contribution, where we look at the *social* phase of the explanatory process; we focus on the construction of explanations that contain both reasons *for* and reasons *against* recommendations. We claim that both kinds of reasons

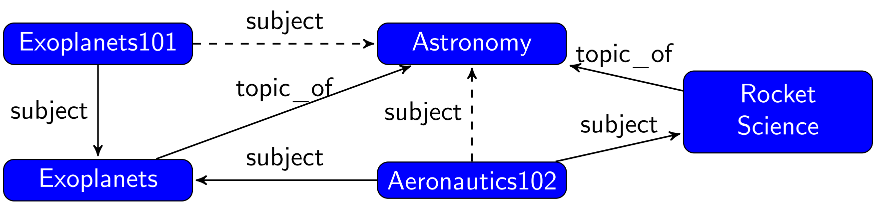



**Figure 3** Sub-graph illustrating reasons for Exoplanets101 and Aeronautics102. For example, the path Exoplanets101 $\xrightarrow{subject}$ Exoplanets $\xrightarrow{topic\_of}$ Astronomy is a reason for Exoplanets101.

must be communicated to users so that they can appropriately trust recommendations and decide what is in their best interest.

We thus focus on the main technical challenge in this work: how to generate reasons *against* a particular recommendation. To do so, we resort to the literature on practical reasoning in Philosophy. There, we find Snedegar's rather comprehensive theory of reasoning [18], a philosophical study of the nature of reasons for and reasons against given options.

Snedegar classifies five schemes through which an agent contemplating competing options may build reasons *against* options. We give a summary of these views below, which we refer to as Schemes S1, S2, S3, S4, and S5.

1. (S1): a reason against an item $A$ is a reason for a competing option;
2. (S2): a reason against an item $A$ is only a reason for *not* $A$ (and not for any particular other option);
3. (S3): a reason against an item $A$ is a reason for the disjunction of alternatives.
4. (S4): a reason against an item $A$ is a reason for the conjunction of alternatives.
5. (S5): a reason against an item $A$ explains (or is part of the explanation as to) why $A$ promotes or respects some objective less well than some other option.[2]

Snedegar has defined these schemes at a highly abstract level, so much so that he distinguishes S1 and S3, a distinction that in our instantiation is not relevant. For us to succeed in bringing those ideas to the realm of CRSs, we must translate them to a more concrete form. We do so by discussing specific instantiations in the remainder of this section.

Our instantiation of S1 generates a reason against a given item by generating reasons for other possible items. For instance, take the case where the system has recommended two classes for someone wishing to learn about Astronomy — Exoplanets101 and Aeronautics102 (Figure 3). A reason against Exoplanets101 would be that Aeronautics102 is about Rocket Science. The intuition behind S1 is similar to the concept of opportunity cost. If one chooses Exoplanets101 instead of Aeronautics102 they will miss the opportunity to learn (or learn less) about rocket science.

Suppose, then, that a system suggests item $e_t$ as a recommendation related to entity $e_h$. We define as $\gamma$ the function that starts with the knowledge embedding parameters $\Theta$ and the path $\pi$, takes inputs $e_h$ and $e_t$, and returns a set of reasons for the recommendation of $e_t$ for the topic $e_h$. The function $\gamma$ represents the cognitive

[2]This scheme requires specifying a quantitative objective.

process of an explanation, and, in the context of this work, it is produced by the DFS-style algorithm in Section 3.

Scheme S2 is rather obvious in real life (a reason against buying a particular product may be that it is known to cause some damage to the environment). However, Scheme S2 does not seem relevant in our context, as we usually have connections among users and items, not reasons specifically against products. We decided not to explore this scheme any further.

Our instantiation of S3 simplifies Snedegar's scheme; it goes through all competing options, collecting reasons for them that are not reasons for the option of interest. In our running example, we can imagine there is a third recommended course, Astrobiology101, and as reasons against Exoplanets101 we have that Aeronautics102, Astrobiology101, or both of them are about rocket science. But we will only present two options at a time, as noted above. In our approach, both S1 and S3 produce identical reasons against.

The instantiation of S4 is similar in spirit to that of S3 to the extent that S4 takes reasons for all competing options into account (reasons against according to S4 are also reasons against according to S3). An example of a reason against Exoplanets101 using S4 would be that necessarily **both** Astrobiology101 and Aeronautics102 from the example above are about rocket science. The stringent nature of this scheme, where the intersection of reasons is required, makes it hard to generate reasons against in the practical circumstances where an CRS may be used.

Algorithms for S1, S3 and S4 can be easily derived from their descriptions. Suppose a CRS recommends $N$ items in an ordered set $\mathcal{I} : \{i_1, i_2, ...i_N\}$ to user $u$. In Scheme S1 (and S3) we define as reason against an item $i_r$ as the union of reasons for each of its alternatives $\mathcal{I}\backslash\{i_r\}$ that are not reasons for $i_r$ itself. Hence we must iterate over the alternatives, extracting reasons for each one of them $\Phi \leftarrow \Phi \cup \Phi_{u,i} \, \forall i \in \mathcal{I}\backslash\{i_r\}$. Note that at this point we assume that the function representing the cognitive process of explanation, $\gamma$ as described earlier in this section, is available. We then remove from $\Phi$ the reasons for our recommendation of interest, if any. The remaining reasons $\Omega = \Phi/\Phi_{u,i_r}$ are the reasons against $i_r$. as presented in the Algorithm 1. Regarding the implementation of Scheme 4 (S4), we follow a similar procedure, except that we take the intersection instead of considering the union of reasons for its alternatives. That is, we just replace the line 13 of Algorithm 1 so as to take the intersection of sets $\Phi \leftarrow \Phi \cap \Phi_{u,i} \, \forall i \in \mathcal{I}\backslash\{i_r\}$.

Figure 4 illustrates the differences between S1, S3, and S4.

Scheme S5 is significantly more complex than the previous ones; we devote the remainder of this section to it. Scheme S5 differs from the other schemes in that it depends on a quantitative objective as the basis of explanations. This objective is used to determine whether a reason is for or against an item. Consider in our toy example that the user has the objective of learning about Exoplanets. With that piece of information, the CRS can present the user with the reason against choosing Aeronautics102 because it is less related to Exoplanets than Exoplanets101 even though Aeronautics102 addresses the subject marginally.

Consequently, the implementation of S5 is quite different from the other schemes. While all the other schemes are implemented by modeling reasons as an unweighted

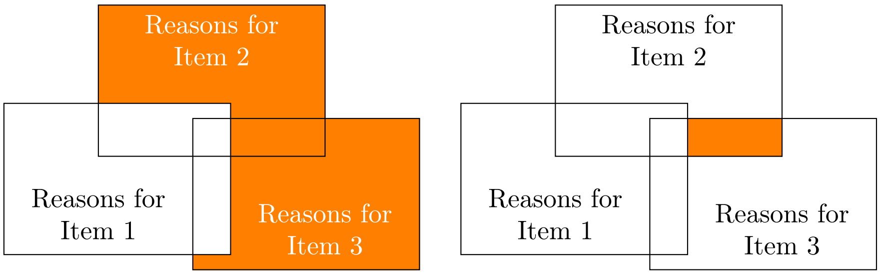



**Figure 4** Suppose we have three items; each rectangle contains reasons *for* an item. Left: according to S1 and S3, the reasons *against* item 1 are all reasons *for* the other items that are not reasons *for* item 1; they are painted orange. Right: according to S4, the reasons *against* item 1 must be reasons that are *for* both the other two items but that are not *for* item 1; again, they are painted orange.

**Algorithm 1** Explanation Generation using Scheme S1

**procedure** REASONS-FOR($i_r$: rec. item, $u$: user, $\Pi$: paths set, $\times$: parameters)
  $\Phi_{u,i_r} = \{\}$ ▷ Set of reasons for $i$
  **for all** $\pi \in \Pi$ **do**
    $\phi \leftarrow \gamma(u, i_r, \pi \mid \mathcal{G})$ ▷ Function describing the cognitive process
    $\Phi_{u,i_r} \leftarrow \Phi_{u,i_r} \cup \phi$
  **end for**
  **return** $\Phi_{u,i_r}$
**end procedure**
**procedure** REASONS-AGAINST-S1($i_r$: rec. item, $u$: user, $\mathcal{I}$: rec. set, $\Pi$: paths set)
  $\Omega_{u,i_r} \leftarrow \{\}$ ▷ Set of reasons against $i_r$
  $\Phi = \{\}$
  $\Phi_{u,i_r} \leftarrow$ REASONS-FOR($i_r$, $u$, $\Pi$, $\mathcal{G}$) ▷ Set of reasons for $i_r$
  **for** $i \in \mathcal{I} \backslash \{i_r\}$ **do** ▷ Iterate over $i_r$ alternatives
    $\Phi_{u,i} \leftarrow$ REASONS-FOR($i$, $u$, $\Pi$, $\mathcal{G}$)
    $\Phi \leftarrow \Phi \cup \Phi_{u,i}$
  **end for**
  $\Omega_{u,i_r} \leftarrow \Phi \backslash \Phi_{u,i_r}$
  **return** $\Omega_{u,i_r}$
**end procedure**

directed graph, in Scheme S5 we must consider weights. To understand this, consider again our toy example in Figure 3. If we look at reasons and disregard weights, we are limited to categorical comparisons, i.e. one can only tell "Aeronautics102 teaches Rocket Science, but Exoplanets101 does not". This limitation imposes difficulties when we need more granular information to compare two recommended items. For example, we can tell that both Exoplanets101 and Aeronautics102 are about Exoplanets, but we cannot compare them since we do not know how strong the link relates each course to its subjects. On the other hand, if we model relations with weights, we can capture that "Exoplanets101 is more related to Exoplanets than Aeronautics102". Therefore, using a more sophisticated weighted directed graph model to represent our reasons, we can leverage the expressiveness of our explanations while improving coverage and support.

We propose to use the plausibility scores from the knowledge embedding itself to rank the recommended entities according to the quantitative objective available beforehand. While the objectives of users are hard to guess, we assume that in our context users pursue strong links connecting their recommended items to their preferences. Thus, in our explanatory framework, the "available objective" for Scheme S5 takes the form of paths in the graph, while the embedding plausibility score measures how well each path meets the user goal. For instance, in our example where students, who want to learn about Astronomy, finds themselves choosing between two courses: Exoplanets101 and Aeronautics102. We know both courses are about Exoplanets, which is a topic of Astronomy, and we know that the fact that a course is about a topic of Astronomy is a compelling reason for choosing the particular course. In this example, our proposal for Scheme S5 takes that the "available objective" is to learn about topics of Astronomy and that the predicted strength of the links connecting a course and topics of Astronomy (the embedding plausibility score) is the measure of how well each course meets this objective.

So, firstly, we collect all the reasons for the recommended item of interest, $i_r$, and we rank $i_r$ for each reason according to the embedding plausibility score. Secondly, we iterate over all the alternatives to $i_r$ repeating the same procedure we described. If an alternative is better ranked than $i_r$, this alternative is more related to that reason than $i_r$, so it is a reason against $i_r$. The whole procedure is described in Algorithm 2.

# 5 Experiments

Our work focuses on improving the interpretability of conversational recommendation systems through novel approaches for explanation generation. We have developed a system that recommends classes offered by the University of São Paulo (USP) to undergraduates so as to test our proposals. Our recommendation system is developed upon a real-world large-scale knowledge graph called USPedia that we have also built in this work [40]. The recommendation task consists in recommending the top three classes given a topic of interest informed by the user. For instance, if the user asks for classes related to subject Astronomy, our CRS should output a ranking of the three most related classes.

We evaluate interpretability at three different levels: the functional, the human and the application ones [23]. These levels grow in complexity and implementation cost from the first to the last one (Figure 5). The functional level evaluates the explanation in objective terms based on a proxy task and, thus, does not require human subjects. The functional level alone is not appropriate to test the subjectivity of explanations, as that relies on the perception of users. The human level provides insights about user perception, and the application level tests the system as a whole and, thus, considers practical aspects, such as response delays, in the evaluation. We highlight in the following sections some of the limitations related to each evaluation level in the context of our work and discuss approaches from the literature to mitigate the observed drawbacks [44, 45].

**Algorithm 2** Explanation Generation using Scheme S5

**procedure** RANK($\Phi$: reasons set, $u$: user, $\Theta$: parameters)
  $r = \phi_i \in \Phi$ ▷ The quantitative objective entity
  $\rho \leftarrow f_r(u, i \mid \mathcal{G})$
  $\Psi \leftarrow$ sort_desc($\Phi$, by=$\rho$) ▷ Sort by the embedding plausibility score
  **return** $\Psi$
**end procedure**
**procedure** SCHEME-S5($i_r$: rec. item, $u$: user, $\mathcal{I}$: rec. set, $\Pi$: paths, $\mathcal{G}$: parameters)
  $\Omega_{u,i_r} \leftarrow \{\}$ ▷ Set of reasons against $i_r$
  $\Phi = \{\}$
  $\Phi_{u,i_r} \leftarrow$ REASONS-FOR($i_r$, $u$, $\Pi$, $\mathcal{G}$) ▷ Set of reasons for $i_r$
  $\Psi_{i_r} \leftarrow$ RANK($\Phi_{u,i_r}, u, \mathcal{G}$)
  **for** $i \in \mathcal{I} \backslash \{i_r\}$ **do** ▷ Iterate over $i_r$ alternatives
    $\Phi_{u,i} \leftarrow$ REASONS-FOR($i$, $u$, $\Pi$, $\mathcal{G}$)
    $\Psi_i \leftarrow$ RANK($\Phi_{u,i}, u, \mathcal{G}$) ▷ Reason against $i_r$
    **if** ($\Psi_{i_r} < \Psi_i$) **then** ▷ Compare if $i$ better meets the objective than $i_r$
      $\Omega_{u,i_r} \leftarrow \Omega_{u,i_r} \cup (\Phi_{u,i} \backslash \Phi_{u,i_r})$
    **end if**
  **end for**
  **return** $\Omega_{u,i_r}$
**end procedure**

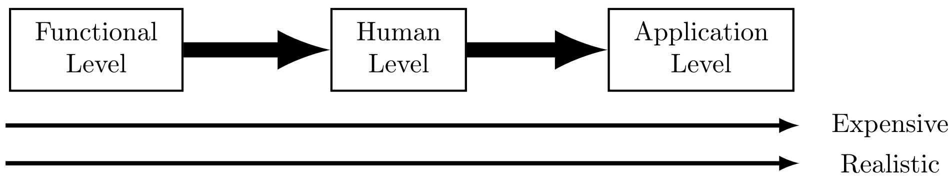



**Figure 5** The three levels at which we evaluate the interpretability of our proposals.

We employed functional level tests to evaluate the feasibility of our proposals in practical settings. We moved forward by resorting to surveys with users where we asked them to compare examples from different explanation methods. These examples were simplified interactions with the real application. Additionally, we have carried out a preliminary application level evaluation. We integrated the most promising explanation methods into versions of the real CRS that were then compared in a survey with real users from our application domain.

We examine each one of these evaluation procedures and the metrics we used in the next three subsections; we explain the set-up in the final subsection.

## 5.1 Functional Level

A CRS cannot impose long delays on its users; in addition, an adequate interpretability method should be able to explain most if not all the recommendations. None of these characteristics rely on human subjectivity and, thus, they can be evaluated at a functional level.

We define the following three metrics to evaluate at this level:

1. *Coverage or Recall*: The fraction or percentage of recommended items for which the interpretability method can find at least one explanation.
2. *Support*: The average number of explanations the interpretability method can find for each recommended item.
3. *Response or Execution Time*: The average time the interpretability method takes to find explanations for each recommended item.

The main goal of our functional level is to evaluate the feasibility of an interpretability method. Therefore, we want to verify whether the method achieves high coverage while coping with the strong execution time constraints of interactive applications. Support is an auxiliary metric that tells us how the method scales in scenarios where multiple explanations are required.

To realize our functional level evaluation, firstly, we built a data-set of simulated user interactions from 10% randomly selected entities of USPedia [40], where each entity represents a subject that one student could ask class recommendations for. We collected the recommendations for each one of the sampled entities. Next, we ran each interpretability method in an offline manner and recorded the metrics of interest.

Our functional level evaluation has the following limitations and caveats:

- **Domain Generalization**: We carried out all experiments in the single application domain of class recommendation, so we cannot ascertain the generalization of our methods toward other domains.
- **Simulation Generalization**: We built simulations randomly selecting entities from USPedia, so we did not account for any popularity bias that may appear in real-world scenarios. It is reasonable to expect that students will ask for recommendations of some popular topics more often than others. Thus, we could observe different results in applications if our methods perform poorly for the most popular subjects.

## 5.2 Human Level

The human-level evaluation is intended to offer preliminary insights or early validation of an interpretability method from the user perspective. To translate the subjective user perception into quantifiable measures, we adopt the following five *explanation aims* [1]: transparency, trust, persuasion, engagement, and effectiveness. Each one of them represents a particular goal of an explanation and is commonly used as a benchmark for recommendation systems [1, 10].

The evaluation was carried out with undergraduate engineering students at the University of São Paulo (USP). All students ranked each interpretability method with respect to the explanation metrics using a survey-based Likert psychometric scale [46] from 1 to 5 (standing for "Strongly disagree", 2 "Disagree", 3 "Neither agree nor disagree", 4 "Agree", and 5 "Strongly agree"). We used this scale to reduce central tendency and social desirability biases where participants do not want to be identified with extreme positions. Finally, each volunteer could write a short, free text with thoughts about the methods they had just evaluated.

In the remainder of this section we discuss the limitations and strengths of our human-level evaluation other than the ones already discussed in the previous section:

- **Sample Generalization**: We used a few handpicked recommendation examples to build the surveys used in our human-level evaluation, so we cannot ascertain how our proposed methods would generalize for different ones.
- **Demographic Generalization**: All surveys were targeted toward engineering students from USP, so our results are still very limited to a young, male community from southeast Brazil.
- **History and Maturation**: To minimize the effects of time passage, we ensured that all participants had to complete the survey in one go.
- **Instrumentation**: To minimize the impact of learning effects on our results, i.e. participants improving over repetition, we randomized the order of questions they were asked to answer. Thus, we expect an overestimation due to learning effects but equally distributed, allowing relative comparisons.
- **Experimenter Bias**: To minimize any unconscious bias being conveyed to participants, experimenters were only allowed to answer technical questions.
- **Misunderstanding**: To minimize the risk of participants misunderstanding the survey instructions due to lack of clarity, we asked some volunteers, without any contact with the actual participants, to fill the survey. In addition, we required every participant to answer correctly a series of practice questions before moving forward with the experiment.
- **Technical Variance**: To minimize variance due to participants using different hardware and software, we asked in advance for preparing an environment with a stable internet connection and implemented the survey in a Google Form to avoid compatibility issues.
- **Multiple Submissions**: Participants were allowed to answer the survey only once.
- **Selection**: Participants were asked to join in the experiment as an optional rewarded task in a USP engineering class. So, even though the participants had to volunteer, there were incentives in place. Since our results were drawn from this self-selected population, they might not generalize. On the other hand, all participants were potential real users from our application domain, which strengthens our results.
- **Ecological Validity**: The fact that our surveys were carried out remotely, instead of in a physical environment like a laboratory, increases their ecological validity. Since participants were in their usual surroundings, the effects of being in an unfamiliar setting were mitigated.
- **Drop out**: Even though participants were free to drop out of the experiment, this effect was minimized due to the incentive of receiving an additional grade in the USP engineering class.

## 5.3 Application Level

As our target audience, USP undergraduate students, was directly accessible, the application level evaluation was similar to the human level one. However, at the application level, we asked participants to interact with a real recommendation system instead of a mocked scenario. Thus, functional factors such as response time and coverage impacted the user perception evaluation.

Regarding the limitations and caveats related to our application level evaluation, while the concerns regarding sample and simulation generalization are minimized (i.e. users can freely interact with the system as in a real-world scenario), other three issues become more evident. These are learning effects from the order of questions, misunderstanding of the interface, and technical variance from the computer or internet connection they have.

### 5.4 Experiment Set-Up

This research depends on the development of a CRS where our hypothesis can be properly evaluated, including with human volunteers. We developed a complete CRS that recommends classes offered by USP to undergraduate students. Our CRS is built upon a real-world, large-scale knowledge graph called USPedia [40].

Once we constructed our knowledge graph, we trained a TransE [32] knowledge embedding model with 500 dimensions for 1000 epochs. We opted for using a batch size of 500, alpha equal to 0.001, margin 1.0, and the optimizer ADAGRAD to perform the training. We selected TransE because it is commonly used as a benchmark in the literature [12, 31].

Using the trained knowledge embedding, we implemented a neighborhood-based recommendation system. Our recommendation system does entity ranking in the form of $\langle head, relation, ?\rangle$, in which the *head* is a conceptual entity representing a preference of theme provided by the user and *relation* is the subject relationship modeling classes. Therefore, we consider the plausibility score provided by the knowledge embedding to rank entities and, then, realize a Top-3 recommendation following the abstract mechanism described earlier in our proposal.

## 6 Results and Discussion

In this section, we describe the results from our simulated experiments and those with real users. First, we present some anecdotal examples in Section 6.1 that highlight the strengths and weakness of our proposals in the current set-up. Next, we examine the feasibility of our techniques in Section 6.2 and then we discuss the reaction of human users to our approach in Section 6.3.

### 6.1 Anecdotal Examples

To illustrate the explanations generated by our proposals, Figure 6 depicts a real explanation example generated by our method. Entities and relations found in the graph appear in the figure, while the textual explanation derived from them appears in the caption. In this particular case, the system recommended the class titled Legal Engineering to attend the requested preference about History subject. Here, the system explains by arguing that the recommended class is about Law, which is a Humanities topic just like History. This example highlights the ability of our methods to leverage ontological connections to offer a rationale for recommendations. In addition, in this example, the relationship between the class Legal Engineering and Law was inferred by the embedding and was not present in the original knowledge graph due to its inherent incompleteness. The ability to benefit from the embedding to infer missing links, besides increasing coverage, can potentially shed light on which relationships are viewed as important by the embedding when producing recommendations.

On the other hand, the ability to leverage inferred relationships can have drawbacks. As the reasons produced by our explanation methods can employ relationships that are not necessarily grounded on known facts, explanations may resort to false facts, which can be often easily spotted by the end user. When the system is forced to explain a bad recommendation, i.e. a suggestion that poorly fits the preference subject, weak or clearly false relationships tend to arise in the system explanations. For instance, Figure 7 explains the connection between the recommendation of a class titled HeavyConstruction and the subject of interest Medicine by saying that both Medicine and WorkAccident are correlated themes. Despite hurting the chance of a user following the system's choice, we argue that even non-sense explanations serve the purpose of empowering the user to critique and disregard bad recommendations.

## 6.2 Results from Simulated Experiments

In this section we report on functional level experiments that were designed to address the following research questions:

1. Are our explanation schemes feasible from an implementation perspective?
2. Can we find at least one explanation for a greater fraction of recommendations when we search the knowledge embedding than the original graph given timeout constraints?
3. How long does it take to find explanations using the knowledge embedding? Is time-to-response acceptable?

We designed user-simulated experiments to evaluate *coverage* and *support*. We arbitrarily established a timeout constraint of 5 seconds to account for responsiveness requirements of a conversational system.

Regarding *reasons for*, Table 2 shows the overall coverage and support for our proposed DFS-based interpretability method using the knowledge embedding as source for explanations (referred to as PRED) as compared to the baseline using the original incomplete knowledge graph (referred to as TRUE). The results show that we obtained 79.33% coverage and a support mean of 2.0 for the embedding-based approach, compared to 42.3% coverage and 1.8 support in the graph-based. We observe that indeed our proposal of replacing the original knowledge graph by the knowledge embedding improves coverage significantly.

As for *reasons against*, we ran experiments with Schemes S1, S4 and S5; all schemes were implemented with the embedding-based approach. Both the coverage (85.1%) and support (2.3) obtained for S1 (the same for S5) are higher than those from *reasons for*. This result was expected as S1 considers more aggregated reasons for alternatives than it removes from the explained recommendation. On the other hand, Scheme S4 could *not* generate a single reason against at all (coverage 0%!). As Scheme S4 requires

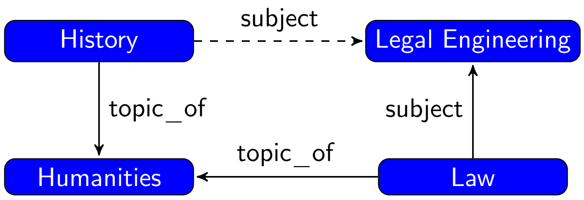



**Figure 6** "LegalEngineering is recommended as it is about Law and both Law and History are topics of Humanities"

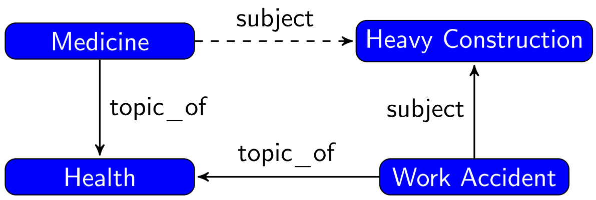



**Figure 7** "HeavyConstruction is recommended as it is about WorkAccident and both WorkAccident and Medicine are topics of Health"

that a reason against an option must be a reason for all of its alternatives, it imposes a restriction so rigorous that it is in fact unfeasible in practice.

Figure 8 presents the behavior of the recall for our *reasons for* proposed method PRED (*embedding recall*) compared to the baseline TRUE (*graph recall*). It also shows the average number of explanations found (*avg. explanation number*) and average execution time (*avg. exec. time*) for our proposed method, when varying time constraints (timeout).

While the baseline method, which uses only the original graph to search for explanations, is by far faster than our proposed method, we observe that the graph recall achieves a certain degree of "saturation" at 42%, which is a significantly lower level than the embedding one at 99%. Here we consider "saturation level" the point where one does not have timeout constraints, i.e., virtually infinite time to search for explanations. Thus, we verify that the original knowledge graph cannot find explanations for less than half of recommendations in our experiment; also, it is not sensitive to time constraints, i.e. saturates into a flat line within milliseconds. On the other hand, the embedding recall, despite having a slow start (close to 0 for timeouts shorter than 2.3 seconds), grows larger than the graph recall for timeouts longer than 3 seconds. Indeed, for a timeout of 5 seconds (that can be considered acceptable for an interactive application), we observe that our proposed method can explain almost two times more recommendations than the original graph alone. This answers our second research question. Note that the average number of explanations and the average execution time behave linearly, for this timeout value. This points out that it may be expensive, in terms of computation cost, to find multiple explanations for the same recommendation.

**Table 2** Coverage and support for *reasons for* using the embedding-based (PRED) and the graph-based approach (TRUE), and reasons against using Schemes S1, S4 and S5. Note the support for each scheme is presented with its respective standard deviation.

| Type | Scheme | Coverage | Support |
|---|---|---|---|
| Reason For | TRUE | 42.3% | $1.8 \pm 1.0$ |
| | PRED | 79.3% | $2.0 \pm 1.0$ |
| Reason Against | S1 | 85.1% | $2.3 \pm 1.4$ |
| | S4 | 0% | - |
| | S5 | 83% | $1.0 \pm 0.5$ |

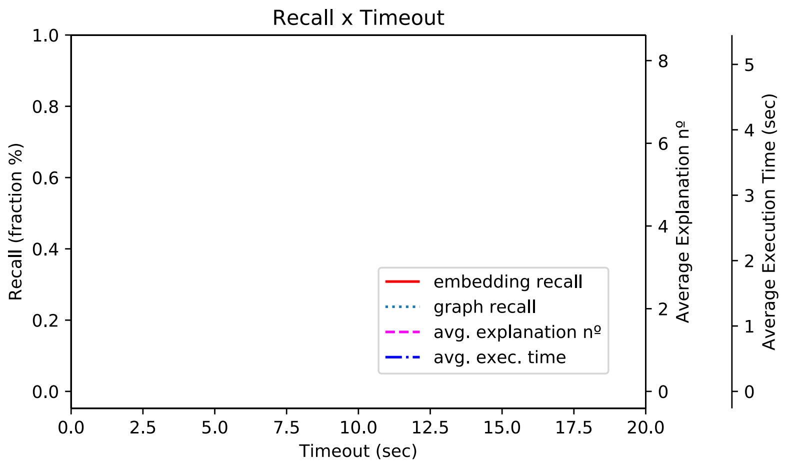



**Figure 8** Comparing recall between our proposal (embedding recall) and the baseline (graph recall). Also we display the average explanation number (avg. explanation no.) and the average execution time (avg. exec. time) for our proposal.

Figure 9 shows the boxplots, with suppressed outliers for better visualization, of the execution time of our proposed method PRED for different numbers of explanations. In this experiment, we aim to evaluate how long it takes to find a given number of explanations for a recommendation. We can observe that all boxplots are skewed down, and the top whiskers are longer than the bottom ones; also, the variability of execution time increases as more explanations are demanded. Considering an acceptable response time (for instance, 5 seconds), for a small number of explanations (one to three), the median value is acceptable, and for a single explanation, even the maximum value is acceptable. Therefore, our approach does produce multiple explanations but not too many of them, answering the third question.

## 6.3 Results from Tests with Human Subjects

We run four user tests. We first run an application level experiment focused on the *reasons for* generation and intended to compare the embedding-based approach (PRED)

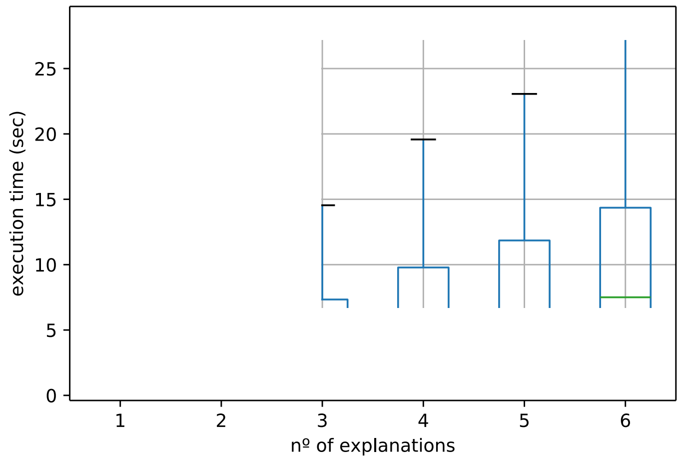



**Figure 9** Execution time of PRED method considering explanation number constraints.

against the graph-based baseline (TRUE). We then run a human level evaluation to validate our hypothesis about the introduction of reasons against in explanations. We run the final application level test where we compared multiple reasons against schemes against the reasons for only baseline. Finally, we run a final human level test to assert the user perception on multiple reasons against schemes.

We present the results for the human level evaluation in Section 6.3.1, and we discuss the two application level tests together in Section 6.3.2.

### 6.3.1 Human Level Evaluation

In this section we report on human-level evaluation experiments that were designed to address the following questions:

1. Do users perceive value in the explanations produced by our schemes?
2. If they do perceive value, which scheme performs best?

We conducted a user study involving 54 subjects from the engineering post-graduate program at USP (88% = male, 78% = born in Brazil's southeast, 72% = $age <= 30$, and 98% = high tech affinity). All demographic data was reported by the subjects.

To run our experiment, we asked the subjects to evaluate explanations generated using our proposed Schemes S1 and S5 for six recommendation instances. The set up was designed to run a block-randomized experiment within subjects. Each user evaluated explanations from both S1 and S5, according to a five point Likert-scale, for three explanation aims: persuasion, trust and transparency, respectively. The order the schemes were presented at each step varied randomly.

Figure 10 depicts the entire experimental set up. Each subject involved in the experiment carried out the following steps:

**(1) Disclosure Agreement**: First, we ask the user to accept a disclosure term granting access of their answers for academic purposes.

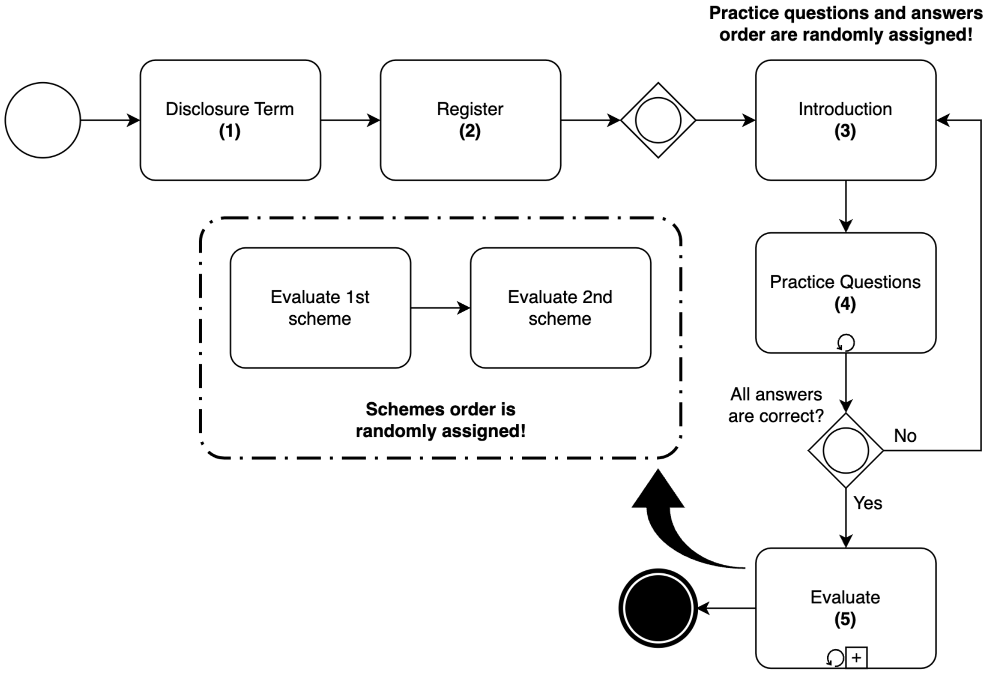



**Figure 10** Diagram representing the human-level experimental set-up we adopted.

**(2) Collection of demographic data**: Subjects were asked to provide common demographic information regarding age, gender self-identification, Brazilian macro region as place of birth and tech affinity, with guarantees on privacy concerning personal data. Our goal was to better understand our sample.

**(3) Introduction and metrics**: We presented all the explanations aim [2] and asked each subject to read them carefully.

**(4) Practice questions**: Subjects were asked to respond multiple answer questions about all the explanation aims. When the user answers all questions correctly, they proceed, otherwise they were asked to go back to step 3 and repeat the questionnaire. This phase guarantees that all subjects understood the experiment purpose and its metrics.

**(5) Explanation scheme evaluation through questionnaire**: Subjects were asked to evaluate S1 and S5 according to the explanation aims: persuasion, trust and transparency, respectively, in a five point Likert-scale. Table 3 contains the details of the questionnaire presented.

Figure 11 presents the arithmetic mean scores obtained from the survey for each explanation aim considering Schemes S1 and S5. The confidence intervals were calculated with bootstrapping at 95% confidence. We can observe that mean scores for all explanation aims were closer to "neutral" or "agree" in the Likert-scale, which indicates that, at least in our specific set up, users perceived value in the explanations.

**Table 3** Questionnaire details.

| Aim | Question |
|---|---|
| persuasion | I feel the explanation persuaded me to follow the recommendation |
| trust | I feel more confidence in the system after receiving the explanation |
| transparency | I feel I better understood the recommendation after receiving the explanation |

In addition, note that scheme S5 ($\mu_{persuasion} = 3.3$ and $\mu_{trust} = 3.83$) received greater mean scores than S1 ($\mu_{persuasion} = 2.48$ and $\mu_{trust} = 3.07$) for persuasion and trust. Note the confidence interval whiskers do not overlap; indeed this difference is statistically significant considering a t-test (the p-value for persuasion is $0.000838 < 0.05$ and the p-value for trust is $0.000971 < 0.05$). On the other hand, for transparency, S1 obtained a greater mean value than S5, however, for this metric we didn't achieved statistical significance (the p-value for transparency is $0.2 > 0.05$). All average values are summarized in Table 4. These results indicate that S5 appears to perform better than S1 in persuasion and trust, while having similar results in transparency.

Note that our experimental set up consists of a human-level evaluation and, thus, fails to take into account the impact of practical circumstances, such as system response time and explanation coverage, in the user perception.

### 6.3.2 Application Level Evaluation

In this section we report on experiments that were designed to provide a first glance on the following research questions:

1. Does the quality of the explanations found using the knowledge embedding deteriorate when compared to those using the original graph in a real application scenario?
2. Do reasons for/against have value for users in a real application scenario? If yes, which scheme performs best?

As described previously, in our first application level experiment (focused on *reasons for* only), we produced two conversational recommendation systems, one with the automatically generated knowledge graph as a source of explanations, and the other with our proposed (embeddings-based search) method as a source of explanations. Our goal was to compare both techniques.

We conducted a user study involving 26 undergraduate engineering students from USP, in which each user evaluated two systems, one employing our proposed method and the other one using the original knowledge graph as a source for explanations. The users were asked to evaluate the five explanation aims (summarized in Table 5) using a Likert psychometric scale from 1 to 5 [46]. One interaction consisted of the user asking for a recommendation for 5 different subjects, so we collected a total of 130 interactions.

Figure 12 depicts the whole experimental set up. Each subject executed the following steps:

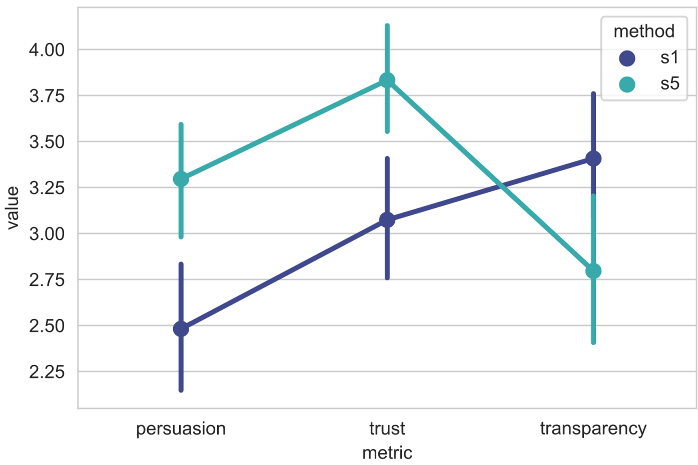



**Figure 11** Visual representation for explanation metrics arithmetic mean scores.

**Table 4** Arithmetic mean scores for explanations from our user study. The highest scores with statistical significance for each metric are highlighted in bold. Statistical significance was assessed by t-tests, with $p < 0.05$.

| Scheme | Persuasion | Trust | Transparency |
|---|---|---|---|
| S1 | 2.48 | 3.07 | 3.41 |
| S5 | **3.3** | **3.83** | 2.8 |

**Table 5** Questionnaire details.

| Aim | Question |
|---|---|
| transparency | Did the explanation help you understand the recommendation? |
| persuasion | On the basis of the explanation, would you follow the recommendation? |
| engagement | Did the explanation have a pedagogical effect? |
| trust | Did the explanation contribute to increase your confidence in the recommendation system? |
| effectiveness | Did the explanation sound coherent? |

**(1) Disclosure Agreement**: First, we ask the users to sign a term accepting to disclose their data for academic purposes.

**(2) Introduction**: Next, we present a detailed description of each explanation aim to be used as evaluation metric. Also, we allow the user to explore a conversational recommendation system without explanation facilities as warm up and to get familiar with the interface.

**(3) Evaluation**: Finally, subjects start the evaluation by asking three recommendations and their respective explanations, at the end of the interaction, the user

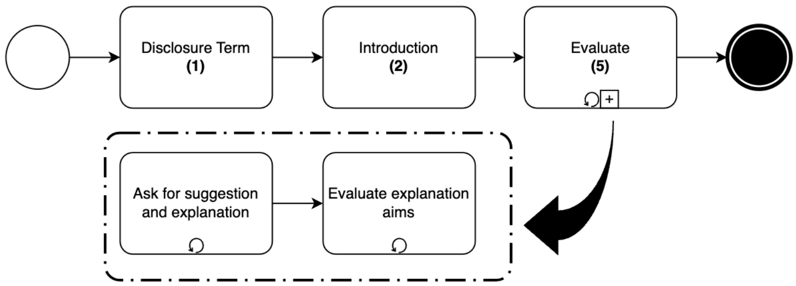



**Figure 12** Diagram representing the application-level experimental set up we adopted.

evaluates all explanation aims. Note the experiment is within subjects and each scheme is evaluated in order. Thus, this evaluation does not account for learning effects and can, potentially, lead to positive bias for the later schemes. This step is repeated for every scheme.

We should emphasize that this experiment had a small sample size, and in this case we did not collect demographic data.

Keeping in mind the exploratory nature of the survey, we now describe the performance indicators from the users' interaction with the conversational recommendation systems. Table 6 presents the arithmetic mean scores provided by the students in our user study for each one of the explanation aims. The upper half of the Table 6 contains the scores acquired in our first application level experiment, where we compared the embedding-based explanations to the graph-based ones. Figure 13 depicts these scores on a continuum representing visually the scale. Comparing both algorithms' overall mean, the knowledge embedding approach (PRED) was better from the user's perspective, $\mu = 2.7$ corresponding to the "neutral" evaluation at the Likert scale.

On the other hand, for the graph approach (TRUE) $\mu = 2.21$ is closer to "Disagree" in the Likert scale. Taking the variable in isolation, effectiveness got the highest average value for both $\mu_{pred} = 2.92$ and $\mu_{true} = 2.64$. This signals that users perceived the explanations as coherent. The TRUE approach had a bad evaluation when the trust was at stake ($\mu_{trust} = 1.92$). As TRUE suffers from knowledge graph incompleteness, it cannot posit explanations for every suggestion. When compared to a better performance of the knowledge embedding approach ($\mu_{trust} = 2.52$), we might conjecture that users prefer any explanation instead of no explanations at all.

The second half of Table 6 presents the scores from our second application level test (focused on both *reasons for/ against*). In this second experiment, we asked 35 undergraduate engineering students from USP to evaluate three conversational recommendation systems, one producing only reasons for as explanations (PRED) and the other two with both reasons for and reasons against (one using S1 scheme and the other S5 scheme). The second experiment was carried out using the same questionnaire (see Table 5) and evaluation procedures as the first one.

The second experiment was focused on comparing multiple techniques of reasons against generation (S1 and S5), and evaluating the presence of reasons against in an explanation against the reasons for only baseline. Note that the baseline PRED was

**Table 6** Average scores for explanation aims from our user study.

| Algorithm | Transp. | Persuasion | Engag. | Trust | Effect. |
|---|---|---|---|---|---|
| TRUE | 2.21 | 2.36 | 2.17 | 1.92 | 2.64 |
| PRED | 2.92 | 2.28 | 2.84 | 2.52 | 2.92 |
| RF | 2.87 | 2.41 | 2.87 | 2.58 | 2.96 |
| S1 | 2.68 | 2.50 | 2.18 | 2.59 | 2.40 |
| S5 | 2.94 | 2.65 | 2.68 | 2.68 | 2.74 |

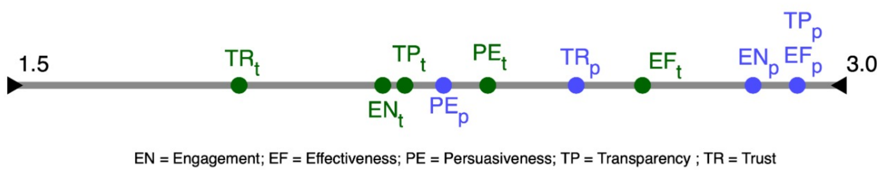



**Figure 13** Visual representation for explanation aims average scores in our first experiment. TRUE and PRED results are in green and blue, respectively.

the same embedding-based method as in the first experiment; indeed, we can observe that the mean arithmetic scores for PRED method in both experiments are similar. Figure 14 depicts these scores on a continuum representing visually the scale. We observe that the systems employing scheme S5 approach has the lead from the user's perspective in terms of transparency ($\mu_{transparency} = 2.94$), trust ($\mu_{trust} = 2.68$) and, notably, persuasion ($\mu_{persuasion} = 2.65$). Furthermore, the scheme S5 appears to be better than S1 in all the evaluated metrics.

As a side effect of introducing reasons against in explanations, we observe a drop in engagement and effectiveness, i.e. the baseline (PRED) achieved the highest scores in both metrics ($\mu_{effectiveness} = 2.96$ and $\mu_{engagement} = 2.87$).

If we compare these results with the ones we obtained in the human level evaluation, from Section 6.3.1, improvements in trust and persuasion were observed, however, the drop in effectiveness and engagement was unexpected. We suppose the bad performance in effectiveness and engagement is due to the complexity overhead added in explanations, i.e. explanations with both reasons for and against are harder to grasp than with reasons for only.

These results serve as initial and exploratory analysis to drive further research focused on properly evaluating these explanation schemes in real-world applications; note that neither attained p-values smaller than 0.05.

# 7 Conclusion

In this paper we have proposed techniques for explanation generation in conversational recommendation systems. Our proposed methods benefit both from knowledge graphs and from related embeddings; our main novel contribution is an emphasis on "reasons

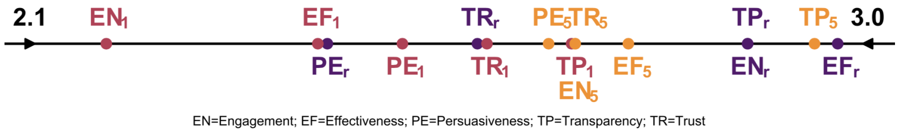



**Figure 14** Visual representation for explanation aims average scores in our second experiment. PRED, S1 and S5 results are in purple, red and yellow, respectively.

against" to improve the user experience when interacting with a conversational recommendation system. First, we proposed a high-coverage explanations method so as to enable a recommendation system to explain its decisions within reasonable time limits, aiming at the cognitive phase of the explanatory process. Aiming at the social phase, we exploited Snedegar's schemes, in particular the scheme S5, to improve the quality of explanations by presenting reasons for and against items. We presented experiments that show our methods to be valuable in practice.

The main limitation of this work is that techniques have been tested in a particular domain; namely, the class recommendation domain. The extent to which our results are valid to a variety of other practical domains must be examined; future work should include testing our methods in other application domains. Moreover, future work should explore other embedding models, in particular state-of-the-art models that improve on TransE, but also other promising paradigms such as semantic embeddings. Effort should also be directed to study the best ways to communicate explanations to users, balancing conciseness, clarity and other factors that affect the impact of explanations.

# References

[1] Tintarev, N., Masthoff, J.: A survey of explanations in recommender systems. In: Proceedings of the 2007 IEEE 23rd International Conference on Data Engineering Workshop. ICDEW '07, pp. 801–810. IEEE Computer Society, Washington, DC, USA (2007). https://doi.org/10.1109/ICDEW.2007.4401070 . http://dx.doi.org/10.1109/ICDEW.2007.4401070

[2] Tintarev, N.: Explanations of recommendations. In: Proceedings of the 2007 ACM Conference on Recommender Systems. RecSys '07, pp. 203–206. Association for Computing Machinery, New York, NY, USA (2007). https://doi.org/10.1145/1297231.1297275 . https://doi.org/10.1145/1297231.1297275

[3] Friedrich, G., Zanker, M.: A taxonomy for generating explanations in recommender systems. AI Mag. **32**, 90–98 (2011)

[4] Sinha, R., Swearingen, K.: The role of transparency in recommender systems. In: CHI '02 Extended Abstracts on Human Factors in Computing Systems. CHI EA '02, pp. 830–831. Association for Computing Machinery, New York, NY, USA (2002). https://doi.org/10.1145/506443.506619 . https://doi.org/10.1145/506443.506619

[5] Jannach, D., Manzoor, A., Cai, W., Chen, L.: A Survey on Conversational Recommender Systems. arXiv e-prints, 2004–00646 (2020) [cs.HC]

[6] Hoeve, M., Schuth, A., Odijk, D., Rijke, M.: Faithfully explaining rankings in a news recommender system. CoRR **abs/1805.05447** (2018)

[7] Padilha Polleti, G., Gagliardi Cozman, F.: Faithfully explaining predictions of knowledge embeddings. In: Encontro Nacional de Inteligência Artificial (ENIAC), pp. 1–12 (2019)

[8] Lv, Q., Ding, M., Liu, Q., Chen, Y., Feng, W., He, S., Zhou, C., Jiang, J., Dong, Y., Tang, J.: Are we really making much progress? revisiting, benchmarking and refining heterogeneous graph neural networks. KDD '21, pp. 1150–1160. Association for Computing Machinery, New York, NY, USA (2021). https://doi.org/10.1145/3447548.3467350 . https://doi.org/10.1145/3447548.3467350

[9] Hogan, A., Blomqvist, E., Cochez, M., D'amato, C., Melo, G.D., Gutierrez, C., Kirrane, S., Gayo, J.E.L., Navigli, R., Neumaier, S., Ngomo, A.-C.N., Polleres, A., Rashid, S.M., Rula, A., Schmelzeisen, L., Sequeda, J., Staab, S., Zimmermann, A.: Knowledge graphs **54**(4) (2021) https://doi.org/10.1145/3447772

[10] Musto, C., Narducci, F., Lops, P., de Gemmis, M., Semeraro, G.: Linked open data-based explanations for transparent recommender systems. International Journal of Human-Computer Studies **121**, 93–107 (2019) https://doi.org/10.1016/j.ijhcs.2018.03.003 . Advances in Computer-Human Interaction for Recommender Systems

[11] Nickel, M., Murphy, K., Tresp, V., Gabrilovich, E.: A Review of Relational Machine Learning for Knowledge Graphs. cite arxiv:1503.00759Comment: To appear in Proceedings of the IEEE (2015). https://doi.org/10.1109/JPROC.2015.2483592 . http://arxiv.org/abs/1503.00759

[12] Gusmão, A.C., Correia, A.C., De Bona, G., Cozman, F.G.: Interpreting Embedding Models of Knowledge Bases : A Pedagogical Approach. In: 2018 ICML Workshop on Human Interpretability in Machine Learning (WHI 2018), pp. 79–86 (2018)

[13] Ruschel, A., Gusmão, A.C., Polleti, G.P., Cozman, F.G.: Explaining completions produced by embeddings of knowledge graphs. In: Kern-Isberner, G., Ognjanovic, Z. (eds.) Symbolic and Quantitative Approaches to Reasoning with Uncertainty, 15th European Conference, ECSQARU 2019, Belgrade, Serbia, September 18-20, 2019, Proceedings. Lecture Notes in Computer Science, vol. 11726, pp. 324–335. Springer, ??? (2019). https://doi.org/10.1007/978-3-030-29765-7_27 . https://doi.org/10.1007/978-3-030-29765-7_27

[14] He, R., Kang, W.-C., McAuley, J.: Translation-based recommendation. In: Proceedings of the Eleventh ACM Conference on Recommender Systems.

RecSys '17, pp. 161–169. Association for Computing Machinery, New York, NY, USA (2017). https://doi.org/10.1145/3109859.3109882 . https://doi.org/10.1145/3109859.3109882

[15] Biran, O., Cotton, C.V.: Explanation and justification in machine learning: A survey. In: IJCAI-17 Workshop on Explainable AI (XAI), vol. 8, pp. 8–13 (2017)

[16] Anelli, V.W., Bellogín, A., Noia, T.D., Donini, F.M., Paparella, V., Pomo, C.: Adherence and constancy in LIME-RS explanations for recommendation. CoRR **abs/2109.00818** (2021)

[17] Chen, L.: Adaptive tradeoff explanations in conversational recommenders. In: Proceedings of the Third ACM Conference on Recommender Systems. RecSys '09, pp. 225–228. Association for Computing Machinery, New York, NY, USA (2009). https://doi.org/10.1145/1639714.1639754 . https://doi.org/10.1145/1639714.1639754

[18] Snedegar, J.: Reasons for and reasons against. Philosophical Studies **175**(3), 725–743 (2018)

[19] Alshammari, M., Nasraoui, O., Sanders, S.: Mining semantic knowledge graphs to add explainability to black box recommender systems. IEEE Access **7**, 110563–110579 (2019) https://doi.org/10.1109/ACCESS.2019.2934633

[20] Ricci, F., Rokach, L., Shapira, B., Kantor, P.B.: Recommender Systems Handbook, 1st edn. Springer, Berlin, Heidelberg (2010)

[21] Koren, Y., Bell, R., Volinsky, C.: Matrix factorization techniques for recommender systems. Computer **42**(8), 30–37 (2009) https://doi.org/10.1109/MC.2009.263

[22] Peters, M.A.: Platform ontologies, the AI crisis and the ability to hack humans 'An algorithm knows me better than I know myself'. Taylor & Francis (2020)

[23] Doshi-Velez, F., Kim, B.: Towards a rigorous science of interpretable machine learning. arXiv preprint arXiv:1702.08608 (2017)

[24] Miller, T.: Explanation in artificial intelligence: Insights from the social sciences. Artificial Intelligence **267**, 1–38 (2019) https://doi.org/10.1016/j.artint.2018.07.007

[25] Gunning, D., Aha, D.: DARPA's Explainable Artificial Intelligence (XAI) Program. AI Magazine **40**(2), 44–58 (2019) https://doi.org/10.1609/aimag.v40i2.2850

[26] Weidinger, L., Mellor, J., Rauh, M., Griffin, C., Uesato, J., Huang, P., Cheng, M., Glaese, M., Balle, B., Kasirzadeh, A., Kenton, Z., Brown, S., Hawkins, W., Stepleton, T., Biles, C., Birhane, A., Haas, J., Rimell, L., Hendricks, L.A., Isaac,

W., Legassick, S., Irving, G., Gabriel, I.: Ethical and social risks of harm from language models. CoRR **abs/2112.04359** (2021)

[27] Gunning, D., Stefik, M., Choi, J., Miller, T., Stumpf, S., Yang, G.-Z.: Xai: Explainable artificial intelligence. Science Robotics **4**(37), 7120 (2019) https://doi.org/10.1126/scirobotics.aay7120

[28] Ribeiro, M.T., Singh, S., Guestrin, C.: "why should I trust you?": Explaining the predictions of any classifier. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016, pp. 1135–1144 (2016)

[29] Zhang, F., Yuan, N.J., Lian, D., Xie, X., Ma, W.-Y.: Collaborative knowledge base embedding for recommender systems. KDD '16, pp. 353–362. Association for Computing Machinery, New York, NY, USA (2016). https://doi.org/10.1145/2939672.2939673 . https://doi.org/10.1145/2939672.2939673

[30] Wood, D., Lanthaler, M., Cyganiak, R.: RDF 1.1 Concepts and Abstract Syntax (2014). http://www.w3.org/TR/2014/REC-rdf11-concepts-20140225/ Accessed 2015-03-15

[31] Wang, Y., Ruffinelli, D., Gemulla, R., Broscheit, S., Meilicke, C.: On evaluating embedding models for knowledge base completion. In: Proceedings of the 4th Workshop on Representation Learning for NLP (RepL4NLP-2019), pp. 104–112. Association for Computational Linguistics, Florence, Italy (2019). https://www.aclweb.org/anthology/W19-4313

[32] Bordes, A., Usunier, N., Garcia-Durán, A., Weston, J., Yakhnenko, O.: Translating embeddings for modeling multi-relational data. In: Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2. NIPS'13, pp. 2787–2795. Curran Associates Inc., Red Hook, NY, USA (2013)

[33] Henk, V., Vahdati, S., Nayyeri, M., Ali, M., Yazdi, H.S., Lehmann, J.: Metaresearch Recommendations using Knowledge Graph Embeddings (2018)

[34] Kazemi, S.M., Poole, D.: Simple embedding for link prediction in knowledge graphs. In: Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., Garnett, R. (eds.) Advances in Neural Information Processing Systems 31, pp. 4284–4295. Curran Associates, Inc., ??? (2018)

[35] Xie, Q., Ma, X., Dai, Z., Hovy, E.H.: An interpretable knowledge transfer model for knowledge base completion. In: ACL, pp. 950–962 (2017)

[36] Zhang, W., Paudel, B., Zhang, W., Bernstein, A., Chen, H.: Interaction embeddings for prediction and explanation in knowledge graphs. Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining (2019) https://doi.org/10.1145/3289600.3291014

[37] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L.u., Polosukhin, I.: Attention is all you need. In: Guyon, I., Luxburg, U.V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., Garnett, R. (eds.) Advances in Neural Information Processing Systems, vol. 30 (2017)

[38] Gardner, M., Mitchell, T.: Efficient and Expressive Knowledge Base Completion Using Subgraph Feature Extraction. In: Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pp. 1488–1498. Association for Computational Linguistics, ??? (2015). https://doi.org/10.18653/v1/D15-1173

[39] Mauro, N., Hu, Z.F., Ardissono, L.: Justification of recommender systems results: a service-based approach. User Modeling and User-Adapted Interaction, 1–43 (2022)

[40] Polleti, G.P., Munhoz, H.N., Cozman, F.G.: Explanations within conversational recommendation systems: Improving coverage through knowledge graph embedding. In: 2020 AAAI Workshop on Interactive and Conversational Recommendation System, p. 8. AAAI Press, New York City, New York, USA (2020)

[41] Polleti, G.P., Souza, D.L., Cozman, F.: Why should I not follow you? Reasons For and Reasons Against in Responsible Recommender Systems. In: 3rd FAccTRec Workshop: Responsible Recommendation, p. 6 (2020)

[42] Musto, C., Narducci, F., Lops, P., De Gemmis, M., Semeraro, G.: Explod: A framework for explaining recommendations based on the linked open data cloud. In: Proceedings of the 10th ACM Conference on Recommender Systems. RecSys '16, pp. 151–154. Association for Computing Machinery, New York, NY, USA (2016). https://doi.org/10.1145/2959100.2959173 . https://doi.org/10.1145/2959100.2959173

[43] Lao, N., Mitchell, T., Cohen, W.W.: Random walk inference and learning in a large scale knowledge base. In: Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, pp. 529–539. Association for Computational Linguistics, Edinburgh, Scotland, UK. (2011). https://aclanthology.org/D11-1049

[44] Roman, N.T., Piwek, P., Carvalho, A.M.B.R.: A web-based experiment on dialogue summarisation. Technical Report IC-05-05, Institute of Computing, University of Campinas (March 2005)

[45] Roman, N., Piwek, P., Carvalho, A.: A web-experiment on dialogue classification. In: International Joint Conference IBERAMIA/SBIA/SBRN 2006 - 4th Workshop in Information and Human Language Technology (TIL'2006), pp. 1–10 (2006)

[46] Likert, R.: A technique for the measurement of attitudes. Archives of Psychology **140**, 1–55 (1932)