Computer Methods and Programs in Biomedicine 235 (2023) 107536

Contents lists available at ScienceDirect

Computer Methods and Programs in Biomedicine

journal homepage: www.elsevier.com/locate/cmpb

Symbolic knowledge extraction for explainable nutritional recommenders

Matteo Magnini[a], Giovanni Ciatto[a,*], Furkan Cantürk[b], Reyhan Aydoğan[b,c], Andrea Omicini[a]

[a] Department of Computer Science and Engineering (DISI), Alma Mater Studiorum – Università di Bologna, via dell'Università 50, Cesena (FC), 47522, Italy
[b] Department of Computer Science, Özyeğin University, Nisantepe Mah. Orman Sok. No:34-36 Alemdağ, Çekmeköy, Istanbul 34794, Türkiye
[c] Interactive Intelligence, Delft University of Technology, Mekelweg 4, Delft 2628 CD, the Netherlands

ARTICLE INFO

Article history:
Received 16 December 2022
Revised 22 February 2023
Accepted 4 April 2023

Keywords:
Explainable artificial intelligence
Symbolic knowledge extraction
Recommendation systems
Nutrition
Neural networks

ABSTRACT

Background and objective: This paper focuses on nutritional recommendation systems (RS), i.e. AI-powered automatic systems providing users with suggestions about what to eat to pursue their weight/body shape goals. A trade-off among (potentially) conflictual requirements must be taken into account when designing these kinds of systems, there including: (*i*) adherence to experts' prescriptions, (*ii*) adherence to users' tastes and preferences, (*iii*) explainability of the whole recommendation process. Accordingly, in this paper we propose a novel approach to the engineering of nutritional RS, combining machine learning and symbolic knowledge extraction to profile users—hence harmonising the aforementioned requirements.

Methods Our contribution focuses on the data processing workflow. Stemming from neural networks (NN) trained to predict user preferences, we use CART Breiman et al.(1984) to extract symbolic rules in Prolog Körner et al.(2022) form, and we combine them with expert prescriptions brought in similar form. We can then query the resulting symbolic knowledge base via logic solvers, to draw explainable recommendations.

Results Experiments are performed involving a publicly available dataset of 45,723 recipes, plus 12 synthetic datasets about as many imaginary users, and 6 experts' prescriptions. Fully-connected 4-layered NN are trained on those datasets, reaching ∼ 86% test-set accuracy, on average. Extracted rules, in turn, have ∼ 80% fidelity w.r.t. those NN. The resulting recommendation system has a test-set precision of ∼ 74%. The symbolic approach makes it possible to devise *how* the system draws recommendations.

Conclusions Thanks to our approach, intelligent agents may learn users' preferences from data, convert them into symbolic form, and extend them with experts' goal-directed prescriptions. The resulting recommendations are then simultaneously acceptable for the end user and adequate under a nutritional perspective, while the whole process of recommendation generation is made explainable.

© 2023 The Authors. Published by Elsevier B.V.
This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0/)

1. Introduction

Recommendation systems (RS) are being exploited extensively in the nutrition domain to help users reach their body shape, weight, or health goals [3–5]. Roughly speaking, nutritional RS are intelligent computational agents aimed at providing users with suggestions about what to eat, possibly leveraging on (*i*) users' data, e.g. age, weight, height, health conditions, etc.; (*ii*) recipes data, e.g. categorization and composition of meals, both in terms of ingredients, and the nutrients therein contained; or (*iii*) experts' background, synthesising the rationale by which recommendations should help users in reaching their goals.

Notably, user information is commonly either explicitly provided by the users or automatically inferred by the agent while interacting with them; recipes data are widely available on the Web; while experts' background can be available to the system in the form of *prescriptions*. Generally speaking, experts' prescriptions should express the criterion by which meals should be recommended to the users, as a function of which nutrients the meal is composed by, and what the physiological parameters of the user – other than their goals – are. Such a criterion should then be en-

* Corresponding author.
E-mail addresses: matteo.magnini@unibo.it (M. Magnini), giovanni.ciatto@unibo.it (G. Ciatto), furkan.canturk@ozu.edu.tr (F. Cantürk), reyhan.aydogan@ozyegin.edu.tr (R. Aydoğan), andrea.omicini@unibo.it (A. Omicini).

https://doi.org/10.1016/j.cmpb.2023.107536
0169-2607/© 2023 The Authors. Published by Elsevier B.V. This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0/)

coded in algorithmic form and exploited to regulate the behaviour of the recommending agent—by driving its recommendations. For any given user and goal, a prescription may indicate which (sorts of) meals the user should eat, possibly when, and in what quantities.

However, given that human nutrition is deeply entangled with cultural, ethical, and subjective concerns, there are two more factors RS should keep into account to be effective, namely: (*i*) users' tastes and preferences concerning food (cf. Kuo et al. [6]), as well as (*ii*) explainability of the recommendation process (cf. Loh et al. [7]). Both aspects are not *technically* required to generate *healthy* food recommendations, but they increase the chances that the system recommendations are actually followed by the users—hence, they help increasing the overall effectiveness of the system.

So, summarising, several requirements must be taken into account when realising *effective* nutritional RS, there including (*i*) correctness, i.e. adherence to experts' prescriptions, (*ii*) acceptability, i.e. adherence to users' tastes and preferences, and (*iii*) explainability of the whole recommendation process. Food RS may function in disparate ways (see Calvaresi et al. [8]); however, in general, they follow a data-driven approach to the construction of recommendations. Data may come from both users and experts, and it is commonly processed via machine learning (ML), behind the scenes of the system development phase. This implies the final system's behaviour will reflect the information carried by data.

Depending on how the relative (mis-)proportions among users and experts data, several tensions may arise among the aforementioned requirements. In fact, if data prioritises experts' prescriptions over users' preferences, agents may output recommendations which are less likely to be seriously taken into account by the users. Vice versa, there is no guarantee recommendations will remain coherent w.r.t. experts' prescriptions—hence useful in the first place. Furthermore, the reliance on sub-symbolic, data-driven solutions may bring both precision and flexibility to the recommender agent, despite hindering its capability of *explaining* how any given recommendation was formed [7].

Accordingly, to mitigate tensions among the acceptability, correctness, and explainability requirements, in this study, we propose a novel approach to the engineering of agent-oriented nutritional RS. Our approach relies on a twofold assumption. On the one side, users' preferences can easily be mined from data, while handcrafting them would likely introduce biases and diverge from reality. On the other side, experts' prescriptions are commonly handcrafted in schematic form (e.g. tables/schedules with meal patterns), and they would be hard to learn from data—as ML may not be able to capture humans' experience. Under these assumptions, we try to answer the following research question: can we combine such kinds of information to let agents devise recommendations which are simultaneously *correct* (i.e. consistent with experts' prescriptions), *acceptable* (i.e. consistent with users' preferences), and, above all, *explainable* (i.e. for which a motivation can be devised)?

To answer this question, we design a nutritional RS architecture where ordinary ML processing is combined with symbolic knowledge extraction [9,10] (SKE), with the purpose of bringing users' preferences learnt from data in symbolic form. There, SKE is a corpus of techniques aimed at extracting intelligible information out of (otherwise opaque) trained ML predictors. Extracted user's preferences are then combined with experts' prescriptions – which we assume are already in symbolic form – to enable the generation of correct, acceptable, and explainable recommendations. Finally, we report experiments to validate the effectiveness of our approach.

Accordingly, the remainder of this paper is organised as follows. In Section 2, we provide an overview of the state of the art of (*i*) nutritional RS, (*ii*) machine learning, (*iii*) symbolic knowledge extraction, and (*iv*) logic knowledge representation/manipulation. There, we recall the main theoretical notions we rely upon in the rest of the paper. In Section 3, we discuss the modelling and the design of our nutritional RS architecture. This is where we present the main contribution of this paper. Next, in Section 4 we describe the design of our experiments aimed at assessing the aforementioned architecture, and we report their results—which are then discussed in Section 5 Finally, Section 6 concludes the paper.

2. Background

In this section, we provide the necessary background on nutritional recommendation systems (Section 2.1), knowledge extraction (Section 2.2) and logic programming (Section 2.3).

2.1. Nutritional recommendation systems

Eating habits play a crucial role in the well-being of all age ranges that calls attention to develop nutritional RS [3,11,12]. These systems emerge to meet a spectrum of user needs [13] such as diet programs [14], chronic disease management [15], critically ill people treatment [16], allergies [17], life-style choices (e.g. sporty, vegetarian, organic, and halal), and physical activity levels [18]. Each preference domain is represented either by inputting expert knowledge (e.g. daily nutrition intake limits, based on the user's physical activity level) or inputting users' feedback based on reviews on recipes [19] into the system.

In addition to representing user profiles in the system, modality of recommendation items such as food, recipe, and meal is the second aspect of knowledge representation in nutritional RS. Many ingredients are combined to achieve a single recipe which dramatically increases the complexity of nutritional recommender systems. Also, many food attributes like cuisine, category, cooking style and time, and nutrition levels are other factors in users' preferences.

Classical approaches to nutritional recommendation learn users' preferences based on their past activities on the system like ratings, clicks, reviews, and browsing history [20]. Those systems derive a user profile from such data sources through content-based [21,22] and/or collaborative filtering [23], then recommend foods / recipes that match the user profile most or are consumed more frequently by other similar user profiles. Recent studies refer to advanced machine learning techniques to personalise nutritional RS, such as question answering over knowledge bases [22,24], recipe retrieval from visual records of available ingredients [19], and learning recipe representations from multi-modal data (e.g., user reviews, recipe text, and photographs) [25].

Although users can find recipes that are suitable for their taste on the Internet, most of the recipes on food websites are poor in terms of health [26]. Therefore, recent food recommendation studies are dedicated to promoting healthy eating to users alongside considering their food preferences [13,19,27]. Exposing a predefined healthiness indicator attached to recommendations is a primary way of promoting healthy foods [26,28] whereas visual attractiveness of food visuals can be more effective to motivate users to embrace healthier choices [29,30]. One alternative approach to enhance user acceptance of recommendations presented by a system is providing explanations on how/why the system recommends a specific item to the user [31]. Users trust more transparent systems that can present reasoning behind decisions rather than relying on black-box automated decision processes [32]. By the motivation of building explainable intelligent systems, recent studies [33,34] propose explainable approaches to food recommendation systems. Padhiar et al. [33] present an explanation ontology modelling to provide reasonings behind food and diet recommendations for some sort of user questions. To achieve recommendations that are both acceptable and healthy, Yera et al. [34] evaluate different recommendation explanation approaches with and without incorporating nutritional information into the recipes. Further-

more, in Calvaresi et al. [35], explainable nutritional systems are envisioned, leveraging on a multi-agent architecture to reach both personalisation and explainability. Also, our study positions in this emerging research field, which aims to enable the explainability of nutritional recommendation systems.

2.2. Machine learning and knowledge extraction

Artificial intelligence (AI) has been a key enabler of recommendation systems since their very beginning [36]. While decades ago it was common to rely on symbolic, rule-based techniques for developing *expert* systems – i.e. recommendation systems encapsulating the knowledge of domain experts in the form of machine-interpretable rules (cf. Fraser and Turney [16]) –, nowadays more and more recommendation systems are relying on machine learning (ML) [37]. This implies a data-driven approach to the engineering of recommendation systems, where both experts knowledge and users' information, as well as the whole process of recommendation production – are (semi-)automatically learnt from data rather than manually encoded by human beings.

Learning from data is automated via ML predictors, often implying *numeric* processing of data—which in turn enables the detection of fuzzy patterns or statistically-relevant regularities in the data, which predictors can learn to recognise. This is fundamental to support the automatic acquisition of otherwise hard-to-formalise behaviours for computational agents, in the form of trained predictors. The whole process involves a number of steps, broadly grouped in training and inference. During the training phase, data from various sources is selected, pre-processed, possibly combined, and finally fed into the ML predictors of choice [38]—e.g., neural networks, decision trees, support vector machines, or linear models. Predictor-specific training algorithms are then exploited to let the predictor identify key patterns in the data, hence learning them. Along the process, the internal structure of the predictor is tuned to adhere to the data. For this reason, we say the predictor is acquiring knowledge in *sub-symbolic* form Calegari et al. [39]. Since then, the predictor will be able to reproduce the learnt patterns in novel scenarios, by drawing inferences on unseen data. This is called the *inference* phase, in jargon.

Trained predictors in inference phase are commonly exploited behind the scenes of computational agents as the inference mechanism enabling many sorts of behaviours. For instance, if the agent's goal is to draw recommendations, trained predictors may be exploited to construct such recommendations. However, the flexibility of state-of-the-art *sub-symbolic* predictors – such as neural networks – comes at the price of poorly-*interpretable* solutions. This implies the expert human user may hardly understand/predict the behaviour of a trained predictor by observing its structure. The overall result is an obscure/unpredictable behaviour of the agent(s). In health-related scenarios – such as nutrition –, the lack of interpretability may be a no-go. This is because it makes it hard for domain experts to guarantee that predictions (and therefore recommendations) are always adequate, under a health-care perspective. Furthermore, sub-symbolic predictors can hardly be updated with novel knowledge, without restarting their training from scratch—an option which is rarely viable after they reach the inference phase.

Among the possible ways to address such issues, in this paper we focus on *symbolic knowledge extraction* (SKE) from sub-symbolic predictors [9]. There, the keyword "symbolic" refers to the way knowledge is represented. In particular, we consider as symbolic any language that is intelligible and interpretable for both human beings and computational agents. Along this line, SKE is the process of distilling the knowledge a sub-symbolic predictor has grasped from data into symbolic form. This can be exploited to elicit the criterion by which a predictor is drawing predictions, in such a way that a domain expert can validate – and, possibly, alter – it. More generally, SKE enables the *inspection* of the sub-symbolic predictors it is applied to, making it possible for the human designer to figure out how they will behave—and possibly, intervene [10].

More precisely, the main idea behind SKE is to enable the construction of a *symbolic* surrogate model mimicking the behaviour of a given predictor. There, symbols may consist of intelligible knowledge, such as *rule* lists or tree—which agent-based technologies may directly manipulate and exploit. In fact, agent-based technologies are very well suited to interoperate with symbolic ones [40]. In particular, symbolic rules may then be exploited (*i*) to either derive predictions directly – i.e. without the predictor –, (*ii*) to better understand the behaviour of the original predictor – hence deriving *post-hoc* explanations [41] –, or (*iii*) to attain easily editable replacements for sub-symbolic predictors—while retaining the capability of learning from data.

Despite symbolic rules could also be mined from data directly (e.g. Tseng et al. [42]), extracting them from trained predictors may be preferable for several reasons. First, sub-symbolic predictors are commonly more flexible – and, potentially, predictive – when it comes to mine information from data. Also, mining rules directly from data, requires data to be available in the first place—which not always the case, especially in the health-care domain where data may be sensitive. Therefore, extracting knowledge from trained sub-symbolic predictors is a way to maximise predictive performance, while reducing dependency from data.

Notably, SKE has already been applied, to the healthcare domain – e.g., to make early breast cancer prognosis predictions [43] and to help the diagnosis and discrimination among hepatobiliary disorders [44] or other diseases and dysfunctions [45] –, as well as to credit-risk evaluation [46–48], credit card screening [49], intrusion detection systems [50], and keyword extraction [51].

2.3. Logic formulæ and logic programming

Logic formulæ are symbolic ways of representing knowledge. They enable agents to draw inferences – possibly, automatically –, via logic reasoning, with the purpose of deriving novel and useful knowledge from prior one. The 'possibly automatically' part is essential here. In fact, logic formulæ can be produced, understood, and manipulated by both human and computational agents. For example, logic formulæ may express under which conditions a given meal is "good" (under some notion of goodness).

Many possible formalisms may be exploited in practice to express logic formalism—including, but not limited to, first order logic (FOL) [52], description logic, or propositional logic. Differences in logic formalisms mostly lay in how they deal with the expressivity–tractability trade-off [53,54]. Intuitively, the easier it is to express knowledge in any given formalism, the harder it will be to algorithmically treat the formilæ expressed via that formalism. Vice versa, more tractable formalisms, will have lower expressive capabilities, hence making it harder to express knowledge.

In this paper, we focus on Horn logic—that is, a particular subset of FOL coming with a fairly good expressivity–tractability trade-off. In Horn logic, knowledge is expressed via one or more logic *clauses*, i.e. rules of the form

$$p(X_1, \ldots, X_n) \leftarrow p_1(\bar{X}') \wedge \ldots \wedge p_m(\bar{X}'') \tag{1}$$

where $p(X_1, \ldots, X_n)$ is a property involving n entities – represented by as many variables $X_1, \ldots, X_n$ –, which can either be true or false. In particular, the property may be true for those particular entities *if*[1] all the m properties in the right-hand side of the formula (namely, $p_1(\bar{X}'), \ldots, p_m(\bar{X}'')$) are true as well for those entities[2].

[1] the '$\rightarrow$' symbol in Eq. (1) should be read as 'if'

[2] the '$\wedge$' symbol in Eq. (1) is should be read as 'and'

There, the symbols $\bar{X}', \bar{X}'', \ldots$ are used to denote subsets of the variables $X_1, \ldots, X_n$.

The left-hand side of a Horn clause is called 'head' and the right-hand side is called 'body'. So, basically, the rule as a whole is stating under which conditions (i.e. the properties in the body) its head is true. If the body is empty – i.e. if $m \equiv 0$ –, then the head is considered to be true—hence the whole rule is called *fact*.

Of course, in practice, one corpus of knowledge may consist of more than one rule. For instance, properties $p_1, \ldots, p_m$ may in turn be defined by other rules. Furthermore, variables may either be assigned with values – i.e. constants referring to entities from the real world (e.g. numbers, strings, etc.) –, or not—hence representing placeholders from unknown entities. In what follows, we denote constants in `monospace`, variables via *Capital* words, and properties names via *lowercase* words.

It is worth recalling that sets of Horn clauses can be read/manipulated not only by human beings, but also by computational agents. Among the many relevant kinds of admissible manipulations which we may delegate to algorithms there is logic *resolution* [55,56]. Thanks to resolution, a computational agent may algorithmically figure out whether a given property is satisfied by a set of clauses or not. This is the essence of logic programming [57], a branch of computer science studying the theory behind logic *solvers* [58]—i.e. software systems in charge of performing resolution, such as, for instance, Prolog [2].

In theory, if a particular property ϕ can be proven as *deducible* from a set of Horn clauses $\mathcal{K}$, then we write $\mathcal{K} \models \phi$, otherwise we write $\mathcal{K} \not\models \phi$. In practice, such a simple notation involves the invocation of some logic solver software, which must load $\mathcal{K}$, be queried with ϕ, and compute (and return) in which circumstances the property ϕ is true.

3. Methods: engineering nutritional recommendation systems

We design a general architecture for a nutritional recommendation system, aimed at drawing *personalised* dietary recommendations for users willing to pursue specific dietary goals. Figure 1 summarises the main components of our architecture, as well as how data is expected to flow through them. Notably, the architecture achieves personalisation of recommendations by *searching* for recipes laying at the intersection among users preferences and experts prescriptions. In the remainder of this section, we discuss how such searching process operates, and what assumption it relies upon.

As depicted in the figure, our architecture assumes the recommending agents may rely upon three disjoint information sources while construction recommendations, namely: (*i*) the user, (*ii*) the nutrition expert, and (*iii*) the database of recipies. In particular, users are considered as the primary providers of their own preferences. Conversely, experts are in charge of reifying each user's dietary goal into *general* prescriptions. Prescriptions are schematic suggestions about what *patterns* of recipes the user could or should eat to pursue that goal. Finally, the database is necessary to let the agent propose admissible recipes to the users, other than making the agent aware of which ingredients and nutrients recipes are composed of.

In the following paragraphs, we delve into the details of how our architecture requires preferences, prescriptions, and recipes to be represented and manipulated. Next, we focus on the notion of recommendation, and on the algorithmic process leading to its production.

Recipes

Our architecture distinguishes among three relevant data types, namely: recipes, ingredients, and nutrients. Recipes are composed of ingredients, which, in turn, may contain several nutrients. All such data types are *named*, meaning that actual recipes, as well as ingredients and nutrients, are identified by their name. In other words, our architecture does not require tracking how meals are actually prepared, but only what ingredients they are composed of, and in which quantity. By tracking the nominal composition of ingredients in terms of nutrients, we let the agent compute the overall nutritional values corresponding to any given recipe.

Along this line, the recipes' database is the architectural component in charge of storing information about recipes, and their ingredients and nutrients. It must support several sorts of queries, including, but not limited to:

- selecting recipes by the ingredients/nutrients they are composed of,
- selecting recipes having (at least, at max, about) some amounts of ingredients/nutrients,
- selecting the ingredients/nutrients corresponding to a given recipe,
- cluster recipes having similar amounts of nutrients.

Notably, queries of these sorts are required by the other components of the recommender agent, described below.

Users' preferences

Our architecture assumes the user's tastes are not explicitly represented, but rather sub-symbolically encoded into a ML predictor. In other words, the recommender agent has a sub-symbolic component, which is capable to learn users' tastes adaptively, from data. This component assumes data describing the recipes the user likes (or dislikes) are available in appreciable amounts.

Along this line, we let the pair $\langle r, p \rangle$ denote a particular preference from the user. There, $r \in \mathcal{R}$ is a recipe from the set of admissible recipes ($\mathcal{R}$), and $p \in \mathbb{R}$ is an appreciation score, where positive values represent appreciation, negative values represent dislike, and 0 represent neutrality. In particular, the appreciation score encapsulates the user's opinion w.r.t. the recipe. It may synthesise their taste, as well as other factors (e.g., allergies, religious/ethical beliefs).

Data representing the user's preferences is assumed to be collected by the agent while interacting with the user, as part of its ordinary operation—possibly, via smart or wearable devices. In particular, data is exploited by the agent as the training set for its sub-symbolic component aimed in charge of learning the user's tastes. Of course, in doing so, the learning algorithm may also access the ingredients- and nutrients-related information stored into the recipes' database. One key aspect of the sub-symbolic approach is that learning should be *continual* [59,60], in order to keep it adherent w.r.t. the user's preferences—which may evolve over time.

Under such assumptions, the users' preferences are modelled as a function

$$appreciation : \mathcal{R} \to \mathbb{R} \tag{2}$$

aimed at predicting the appreciation the user's appreciation score for any given recipe. In practice, this function is implemented via a ML predictor aimed at generalising the user's data acquired by the recommender agent.

Dietary prescriptions

Dietary prescriptions are structured representations of *what* a given user should eat, and *when*, in order (for the user) to achieve a particular *goal*. They are commonly produced by nutrition experts upon request, and structured around the particular physiological features of the user, other than the expert's background knowledge and experience. For any given prescription, the 'what' part consists of particular ingredients or nutrients the user should assume on a per-meal basis, along with the corresponding quantities. Conversely, the 'when' part indicates the moment of the day the meal should be consumed (e.g., breakfast, lunch, dinner, etc.). Finally, the goal is the long-term effect the expert is expecting to produce onto the body of the user, under the assumption that the

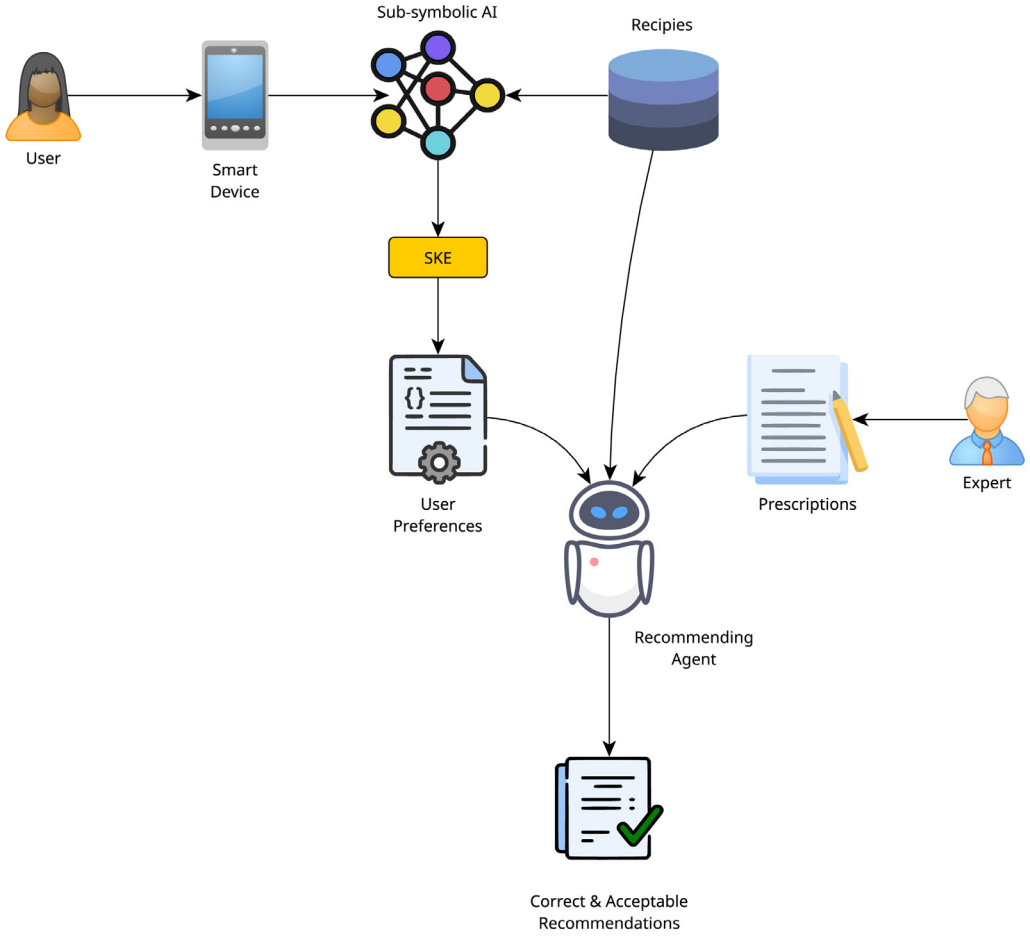

Fig. 1. Anatomy of a nutritional RS under a data-flow perspective. The user interacts – via some smart/wearable device – with a sub-symbolic AI predictor, continuously trained to predict whether the user likes a given recipe or not. A knowledge base is then extracted from the predictor, representing the user's preferences in a human-interpretable logic form. Dietary prescriptions – provided by human experts – are transformed into the same logic form. Conversely, databases providing information about recipies – there including ingredients and their nutrients – are assumed to be remotely available via the Internet. Finally, the recommending agent exploits a logic engine, combining all such information into recommendations which are simultaneously correct (w.r.t. experts prescriptions), acceptable (w.r.t. users' preferences), and explainable.

dietary prescriptions are followed accordingly. The goal should reflect user's request, but it does not need to be explicitly represented in the prescription.

Dietary prescriptions usually come in a quasi-natural language form or in tabular form. In the latter case, each cell represents a particular moment of the week, and the meal therein suggested by the expert. When this is the case, the expert's suggestion may simply indicate the most adequate nutrients/ingredients and their corresponding quantities. It is then the user's responsibility to construct a recipe matching their own tastes—unless, of course, a nutritional RS is in place.

For the sake of simplicity, yet without loss of generality, we focus on the single cell of a dietary prescription table. In other words, we focus on the single prescription concerning any given moment t of the week. There, we assume the prescription consists of one or more *logic formulæ* expressing what properties the user's meal should have to be coherent with the dietary goal.

In the following subsection, we provide details about what logic formulæ are, and how they can express dietary prescriptions.

3.1. The role of logic formulæ

Our architecture leverages upon Horn clauses to represent users' preferences and experts' prescriptions. In fact, as we show in the remainder of this section, Horn logic lets us simply and clearly express dietary prescriptions, while retaining acceptable computational tractability features.

Such formulæ may be written by human beings and exploited by some nutritional RS to suggest actual recipes to eat, or, vice versa, they may be generated by some algorithm, and understood by human beings as computer-generated prescriptions.

More precisely, experts prescriptions at time t consist of a set of Horn clauses defining the *should_eat*/1 predicate. Such predicate intensively describes what recipes the user should eat at time t, by describing admissible (or forbidden) ingredients / nutrients. This is performed via two more predicates – namely, *has*/2 and *has_no*/2 –, which assert what ingredients / nutrients the suggested recipe should or should not be composed by. Groups of ingredients / nutrients may be defined as well, via unary predicates defined by ad-hoc clauses—e.g. *pvegetable*/1.

So, for instance, we may express a particular dietary prescription for Monday at lunch via the following rule:

$$\begin{aligned} \mathit{should\_eat}(R) \quad &\leftarrow \quad \mathit{has}(R, \texttt{rice}) \\ &\wedge \quad \mathit{has}(R, \texttt{chicken}) \\ &\wedge \quad \mathit{has\_no}(R, \texttt{salt}) \\ &\wedge \quad \mathit{has}(R, X) \wedge \mathit{vegetable}(X) \end{aligned} \tag{3}$$

stating that the user should eat *any* meal whose recipe R includes both `rice` and `chicken`, but no `salt`, other than *any* ingredient X which is a vegetable. Of course, to make the set of clauses self-contained, we should also provide rules stating what it means for

a recipe R to have (resp. have no) ingredient X—hence giving semantics to the property $has(R, X)$ (resp. $has\_no(R, X)$). Similarly, we should also provide rules stating what it means for an ingredient X to be a *pvegetable*—hence giving semantics to the property $pvegetable(X)$. For the sake of brevity, we do not report such definitions here.

As probably obvious, encoding recipes into clausal form could be done in other ways, too: for instance, one may include a predicate $pquantity(X, Q)$ stating that ingredient X is present in quantity Q. However, this would not change our core contribution, so for the sake of simplicity we do not discuss the many alternative syntactic choices available here.

Users' preferences can be represented in clausal form as well. In that case, they consist of sets of Horn clauses defining the *likes*/1 predicate. Similarly to prescriptions, such definitions may exploit the predicates *has*/2 and *has_no*/2 too, as well as any other custom predicate defining groups of ingredients / nutrients.

As an example, consider the following clause set:

$$\begin{aligned} likes(R) &\leftarrow has(R, \texttt{rice}) \\ &\wedge\ has(R, \texttt{chicken}) \\ &\wedge\ has\_no(R, \texttt{broccoli}) \\ &\wedge\ has(R, \texttt{peas}) \\ vegetable(\texttt{peas}) &\leftarrow \\ vegetable(\texttt{broccoli}) &\leftarrow \\ has(\texttt{paella}, \texttt{rice}) &\leftarrow \\ has(\texttt{paella}, \texttt{chicken}) &\leftarrow \\ has(\texttt{paella}, \texttt{peas}) &\leftarrow \\ has(\texttt{paella}, \texttt{seafood}) &\leftarrow \end{aligned} \tag{4}$$

It is stating that (*i*) the user likes recipes having `rice`, `chicken`, and `peas` – such as the recipe for `paella` –, but no `broccoli`, (*ii*) that both `peas` and `broccoli` are vegetables, and (*iii*) the composition of `paella`—which also includes `seafood`.

When both prescriptions and preferences are in clausal form, the intersection among them can be computed via logic resolution. This simply requires the query[3]

$$likes(R) \wedge should\_eat(R) \tag{5}$$

to be proved – via logic resolution – against the clause set attained by merging the clausal forms of both prescriptions and preferences. In this way, a logic solver may compute one or more admissible assignments for R – i.e. recipes the user should and would like to eat –, or discover that none exists.

So, for instance, the query in Eq. (5) may be tested against the clause set attained by merging Eqs. (3) and (4). In this way, any logic solver may conclude that $R = \texttt{paella}$. We denote such situation by:

$$(3) \cup (4) \models likes(\texttt{paella}) \wedge should\_eat(\texttt{paella})$$

3.2. The role of symbolic knowledge extraction

Our architecture requires that both users' preferences and experts' prescriptions are available as sets of Horn clauses. Under such assumption, it can construct recommendations via logic resolution.

As far as prescriptions are concerned, we assume that experts can produce them in clausal form, *directly*—or, at least, in forms which can be *automatically* converted into sets of Horn clauses. This assumption is easily met by the current practice of providing prescriptions as timetable of suggested recipes.

Conversely, as far as preferences are concerned, the clausal form requirement is clearly conflicting with Eq. (2), where users' preferences are modelled as trained sub-symbolic predictors. The sub-symbolic representation is adequate, as it enables learning users' preferences from data, and adapting to their change over time. However, such form prevents the direct exploitation of logic resolution as the means to construct recommendation.

Accordingly, to fill the gap, we choose to bring users' preferences in clausal form, algorithmically. To serve this purpose, our architecture leverages upon a SKE step, which is in charge of extracting symbolic knowledge – in clausal form – out of the sub-symbolic predictor, which has been trained to predict users preferences. Again, we do not impose any particular SKE algorithm – meaning that implementers are free to choose the extraction algorithm which is most adequate for their needs –, but we do require the extraction step and we require it to output Horn clauses.

4. Results: evaluation of the approach

To validate our architecture, we here report a number of experiments designed to exemplify and assess a nutritional RS matching our architecture. The source code and the instructions to reproduce our experiments is available at https://github.com/pikalab-unibo/mccao-cmpb-experiments-2022.

More precisely, in our experiments: (*i*) we generate synthetic datasets for a single user's food preferences, (*ii*) we create and train a ML predictor – namely, a neural network – to predict whether a recipe will be liked by the user or not, (*iii*) we apply a SKE algorithm to generate logic rules that describes the internal decision-making behaviour of the predictor, therefore the user's preferences, and, finally, (*iv*) we assess the system capability to recommend recipes that are both compliant to the user's preferences and to nutritional prescriptions. In doing so, we leverage on several datasets, some of which are publicly available from the Web, while others are synthesised as part our experiments.

Details of how we perform data selection and synthesis, and about our experiments are provided in the following subsections.

4.1. Datasets

We use a public dataset of recipes,[4] and 12 synthetic datasets of preferences, for as many synthetic users.

The recipes' dataset consists of four files:

- `01_Recipe_Details` contains all recipes (recipe id, title, source and cuisine);
- `02_Ingredients` contains all basic ingredients (aliased name, synonyms, entity id, category);
- `03_Compound_Ingredients` contains compound ingredients (name, synonyms, entity id, constituent ingredients, category);
- `04_Recipe-Ingredients_Aliases` contains the ingredients for each recipe (recipe id, original ingredient name, aliased ingredient name, entity id).

There are 929 unique basic ingredients and 103 compound ingredients (i.e., ingredients made of multiple basic ingredients or ingredients that can be used instead of a subset of basic ingredients), 1032 in total. Each ingredient belongs to one of the 21 possible categories (e.g., additive, bakery, beverage, etc.). Recipes that have at least one ingredient are 45,749, the ones without any ingredient are ignored (23).

The other datasets that we use in the experiments are synthetic. They represent the preferences of 12 (imaginary) users on

[3] To be read as "is there any recipe R which is simultaneously among the recipes the user likes and should eat?"

[4] Available at https://cosylab.iiitd.edu.in/culinarydb

Table 1
Datasets statistics for each users on test sets. The second column shows the *like* class ratio. Third and fourth columns describe the accuracy and precision score of trained neural networks. Fifth and sixth columns describe the accuracy and precision score of the extracted rules. The last column shows the fidelity of the extracted rules w.r.t. the NN.

users	liked ratio	net acc.	net prec.	r. acc.	r. prec.	r. fid.
user 1	0.336	0.9233	0.8855	0.864	0.8228	0.873
user 2	0.2842	0.9714	0.9564	0.794	0.7688	0.7972
user 3	0.3941	0.8503	0.8221	0.7726	0.7136	0.7955
user 4	0.4502	0.8364	0.8165	0.7562	0.7145	0.7813
user 5	0.3022	0.9621	0.9478	0.8381	0.8301	0.842
user 6	0.2719	0.9722	0.9514	0.7997	0.7109	0.8034
user 7	0.3504	0.8916	0.8369	0.7782	0.7311	0.7901
user 8	0.449	0.7782	0.7594	0.6868	0.674	0.7283
user 9	0.393	0.8085	0.7783	0.7493	0.7524	0.803
user 10	0.4734	0.7381	0.7582	0.6862	0.7452	0.7616
user 11	0.4359	0.7708	0.7863	0.7098	0.7426	0.7804
user 12	0.4353	0.7772	0.7979	0.7047	0.7467	0.7814
average	0.3813	0.8567	0.8414	0.7616	0.7461	0.7948

the 45,749 recipes above. In particular, the *i*th dataset includes a binary label for each recipe – namely, *like* and *dislike* – expressing the preference of user *i* for that recipe. Despite the synthetic datasets are engineered to mimic real human beings, no real personal data is exploited in the process. Hence, the data come with no ethics- or privacy-related concern.

Synthetic datasets are created in two steps. In the first step, we generate unconditional preferences denoting whether or not the user likes each ingredient (e.g., milk) based on predefined users' profiles. A user profile is a set of ranges of values for a certain number of ingredients and/or categories (e.g., vegetable, meat): for instance, an ingredient could have a value between 0 and 10. Each range is used to generate the corresponding value of likelihood with uniform distribution. Associated range for a given category is applied to each ingredient belonging to that category. In the second step, we generate a label regarding the likability of each recipe—i.e., whether the user likes or dislike the recipe. The generation of the label is based on the values of likelihood of the ingredients of the recipe. Details about the synthesis process are provided in Appendix A.1 for the sake of reproducibility.

4.2. Learning users' preferences via sub-symbolic predictors

To learn users' preferences, we rely upon fully-connected neural networks with 1 input layer, 2 hidden layers, and 1 output layers—having 1032, 16, 8, and 1 neurons, respectively. The activation function of all neurons from the input and hidden layers is the rectified linear unit (ReLU), while the activation function of the output neuron is a sigmoid. The network takes a recipe as input – a tensor of 1032 numbers, one for each ingredient – and it outputs a value representing whether the recipe will be liked or not by the user.

It is worth recalling that one particular neural network should be trained for each user. In our case, we train 12 different networks, as we have 12 users/datasets. Each network is trained upon one user-specific dataset using half records (22,874), the remaining are used for the test set (22,875). This test set is also used to evaluate the following steps of the workflow. The training process lasts for 20 epochs with a batch size of 32. At the end of the training, networks are able to reach an average accuracy on the test set of 85.6%. All results are reported in Table 1. We report also the precision of the NN. Precision is computed as the number of true positives over the total amount of positive predictions (true positives plus false positives). This measure is key for systems like this one, where it is more important to identify true positives (actually liked recipes) having few or none negatives (recommended disliked recipes).

4.3. Extracting users' preferences via SKE

Once we have a ML predictor trained to predict if a recipe will be liked or not by a user, we can extract symbolic knowledge from it representing the user's food preferences. Examples of extracted rules for a user are:

$$
\begin{aligned}
\mathit{likes}(R) \quad &\leftarrow \quad \mathit{has\_no}(R, \texttt{egg}) \\
&\wedge \quad \mathit{has\_no}(R, \texttt{pepper}) \\
&\wedge \quad \mathit{has\_no}(R, \texttt{lime}) \\
&\wedge \quad \mathit{has\_no}(R, \texttt{milk}) \\
&\wedge \quad \mathit{has}(R, \texttt{almond}) \\
\neg\mathit{likes}(R) \quad &\leftarrow \quad \mathit{has}(R, \texttt{egg}) \\
&\wedge \quad \mathit{has\_no}(R, \texttt{pepper}) \\
&\wedge \quad \mathit{has}(R, \texttt{parsley})
\end{aligned}
$$

The first rule says that the user likes dishes with almond but without eggs, pepper, lime and milk. The second rule states that the user dislikes dishes with eggs, parsley, and no pepper.

To extract rules, we use a well known SKE algorithm, namely CART [1], that allow us to generate a list of logic rules. CART needs as input a dataset, the one used to train the neural network, and the trained ML predictor that is used as an oracle. Broadly speaking, the undergoing neural network predicts classes for the input dataset. Then, CART is applied to the new dataset (input dataset plus the predicted classes). The output of the algorithm is a classification decision tree (DT) converted into logic rules [61].

The process of transforming a DT into rules is straightforward: each path from the root to a leaf is a rule, where nodes are encoded into logical condition (i.e., if an ingredient is liked or not) and the leaf represent the class assigned to the recipe. The algorithm can take several hyper-parameters, we choose to set the maximum number of leafs – i.e., rules – to $R = 50$ and the maximum depth of the DT to $D = 10$—i.e., the maximum amount of ingredients that can appear in the right hand side of a logic rule.

We choose to constrain the output rules in this way for mainly two reasons. The first one is to put a limit to the growth of the DT avoiding spending too much computational time. If we have N binary features – ingredients – we can obtain a DT with maximum depth of $N + 1$, while the maximum number of leafs – recipes or pattern of recipes – are 2^N. With $N = 1032$ it would be infeasible. The second reason why we limit the DT is that even with fewer ingredients the output rules should not be too long – not including too much ingredients in the right hand side of the formula – and too many — if they are hundreds or thousands no human will read and understand that. Of course this is a deliberative choice that has a trade-off between performance measures (e.g., accuracy, precision) and interpretability. Because in this scenario we are more interested in the explanation of why a prescription may or may not be welcome by a user, we sacrifice performance over interpretability.

Table 1 summarises the accuracy that the extracted rules have over the test sets. It also shows the fidelity values of the rules w.r.t. the sub-symbolic predictors. The fidelity is computed like the accuracy but instead of checking if the prediction is the same as the ground truth of the test set it is compared with the class value predicted by the ML predictor. In other words, the fidelity is an accuracy computed on the test set but with the predicted classes of the networks.

About the choice of decision trees as the means for rule extraction, one may argue that DT are very sensitive to semantic and/or structural *instability* [62,63]. On the one hand, structural instability refers to the situation where small perturbations in the training data lead to considerably different DT. For instance, adding new instances to the training dataset may lead CART to make different splits [64]. This is more likely to happen when some features of the training are highly correlated, hence similarly contributing to

the splitting criteria of some DT training algorithm [65]. On the other hand, semantic instability refers to the situation where perturbations in the training set lead to DT making different predictions.

As far as our approach is concerned, we argue that instability is not a problem. In fact, our methodology does not prescribe any particular means for rule extraction, and CART is one of many possible choices. As long as a high-enough level of fidelity is achieved, other rule-extraction procedures may be adopted instead of CART (cf. Sabbatini et al. [10], Sabbatini and Calegari [66,67], Sabbatini et al. [68]). In particular, here we choose CART because of its simplicity (both in terms of understandability and algorithmic complexity) and scalability.

4.4. Proposed recipes

The goal of our experiments is to understand how users' preferences and nutritional prescriptions are combined to recommend recipes. To do so, we rely upon sets of logic rules expressing domain-expert prescriptions. In total, we have 6 sets of rules, for as many meals—one per meal. For each meal, we may have several (from 2 to 4) rules—one for each dish of the meal. Notably, prescriptions are expressed via the same formalism we adopt to represent user's preferences (see Section 3.1). More precisely, we rely on 6 prescriptions – i.e. three days, two meals per day –, to be shared among our 12 users.

For the sake of readability, we minimise the amount of prescriptions – as well as the complexity of each prescription –, while focussing on demonstrating the feasibility of our approach. As a key simplification, in particular, we omit quantities when representing prescriptions in clausal form. Of course, should our approach be adopted in practice, it would need to take into account more prescriptions, possibly including quantity-related information. However, it is worth mentioning that both the performance and the scalability of our approach are not related to the number of prescriptions the system is taking into account. In fact, prescriptions are provided by experts, whereas recommendations are commonly constructed starting from a single prescription.

Accordingly, in the following we identify the six prescriptions by index, and represent them via the rule sets below:

p.1 First day, lunch: "Rice with vegetables. 80 g of raw rice, 35 g of raw lentils, 120 g of raw chicken, 120 g of mix vegetables. Garlic, herbs, 2 teaspoons of olive oil, 1 piece of orange".

$$
\begin{aligned}
\mathit{should\_eat}(R) &\leftarrow \mathit{has}(R, \texttt{chicken}) \wedge \mathit{has}(R, \texttt{rice}) \\
\mathit{should\_eat}(R) &\leftarrow \mathit{has}(R, \texttt{lentils}) \\
\mathit{should\_eat}(R) &\leftarrow \mathit{has}(R, \texttt{orange}) \\
\mathit{should\_eat}(R) &\leftarrow \mathit{has}(R, \texttt{garlic}) \\
&\wedge \mathit{has}(R, X) \wedge \mathit{has}(R, Y) \wedge \mathit{has}(R, Z) \\
&\wedge \mathit{vegetable}(X) \wedge \mathit{herb}(Y) \wedge \mathit{essential\_oil}(Z)
\end{aligned}
$$

p.2 First day, dinner: "Burger and grilled vegetables. 90 g of beef, 80 g of bread, 120 g of vegetables, 1 teaspoon of oil, 1 cup of strawberries".

$$
\begin{aligned}
\mathit{should\_eat}(R) &\leftarrow \mathit{has}(R, \texttt{beef}) \wedge \mathit{has}(R, \texttt{bread}) \\
\mathit{should\_eat}(R) &\leftarrow \mathit{has}(R, \texttt{strawberry}) \\
\mathit{should\_eat}(R) &\leftarrow \mathit{has}(R, X) \wedge \mathit{has}(R, Y) \\
&\wedge \mathit{vegetable}(X) \wedge \mathit{essential\_oil}(Y)
\end{aligned}
$$

p.3 Second day, lunch: "Tuna salad. 120 g of tuna, 120 g of vegetables, 1 teaspoon olive oil, 80 g of bread, 35 g of raw beans, 1 cup of blueberries".

$$
\begin{aligned}
\mathit{should\_eat}(R) &\leftarrow \mathit{has}(R, \texttt{bread}) \wedge \mathit{has}(R, \texttt{beans}) \\
\mathit{should\_eat}(R) &\leftarrow \mathit{has}(R, \texttt{tuna}) \\
&\wedge \mathit{has}(R, X) \wedge \mathit{has}(R, Y) \\
&\wedge \mathit{vegetable}(X) \wedge \mathit{herb}(X) \wedge \mathit{essential\_oil}(Y) \\
\mathit{should\_eat}(R) &\leftarrow \mathit{has}(R, \texttt{blueberry})
\end{aligned}
$$

p.4 Second day, dinner: "Chicken with mustard and lemon juice. 90 of chicken, 120 g of vegetables, 80 g of raw pasta, 1 teaspoon of olive oil. Mustad and lemon juice, 1 cup of clementines".

$$
\begin{aligned}
\mathit{should\_eat}(R) &\leftarrow \mathit{has}(R, \texttt{chicken}) \wedge \mathit{has}(R, \texttt{mustard}) \\
&\wedge \mathit{has}(R, \texttt{lemon\_juice}) \\
\mathit{should\_eat}(R) &\leftarrow \mathit{has}(R, \texttt{pasta}) \\
&\wedge \mathit{has}(R, X) \wedge \mathit{essential\_oil}(X) \\
\mathit{should\_eat}(R) &\leftarrow \mathit{has}(R, X) \wedge \mathit{vegetable}(X) \\
\mathit{should\_eat}(R) &\leftarrow \mathit{has}(R, \texttt{citrus\_fruits})
\end{aligned}
$$

p.5 Day three, lunch: "Salmon with potatoes. 120 g of salmon, 240 g of cooked potatoes, 120 g of vegetables, 1 teaspoon of butter, 1 pear".

$$
\begin{aligned}
\mathit{should\_eat}(R) &\leftarrow \mathit{has}(R, \texttt{compound\_salmon}) \wedge \mathit{has}(R, \texttt{potato}) \\
\mathit{should\_eat}(R) &\leftarrow \mathit{has}(R, \texttt{pear}) \\
\mathit{should\_eat}(R) &\leftarrow \mathit{has}(R, \texttt{butter}) \\
&\wedge \mathit{has}(R, X) \wedge \mathit{vegetable}(X)
\end{aligned}
$$

p.6 Day three, dinner: "Turkey in papillote. 90 g of turkey, 1 teaspoon of olive oil, 120 g of vegetables, 35 g of raw gram bean, 80 g Raw wholegrain rice, 1 orange".

$$
\begin{aligned}
\mathit{should\_eat}(R) &\leftarrow \mathit{has}(R, \texttt{gram\_bean}) \\
\mathit{should\_eat}(R) &\leftarrow \mathit{has}(R, \texttt{rice}) \\
\mathit{should\_eat}(R) &\leftarrow \mathit{has}(R, \texttt{orange}) \\
\mathit{should\_eat}(R) &\leftarrow \mathit{has}(R, \texttt{turkey}) \\
&\wedge \mathit{has}(R, X) \wedge \mathit{has}(R, Y) \\
&\wedge \mathit{vegetable}(X) \wedge \mathit{essential\_oil}(Y)
\end{aligned}
$$

For each user and for each prescription we compute the recipes to be proposed (consider reading Appendix A.2 for implementation details). To evaluate how the recommendation performs, we calculate the precision over the test sets, i.e., number of proposed recipes actually liked by the user over the total amount of proposed recipes. Results are summarised in Table 2. We remind that the system has never seen recipes in the test sets in any of its phases (NN training, rule extraction and recipes recommendation) except for test itself. Table 3, similarly to Table 2, reports the precision of the recommendation phase but instead of using the extracted rules to evaluate users' preferences it use the sub-symbolic predictors.

It is worth noticing that there is the possibility of corner cases in the recommendation process. If the system is dealing with a user whose preferences are in conflict with the prescriptions it could happen that no recipe is recommended. This scenario is not an issue for our system but it is an intrinsic characteristic of the domain. To avoid empty recommendations the right action to take is simply change prescriptions to be more adequate to user's preferences. In the computing of accuracies these corner cases are ignored.

4.5. Results

In order to keep our experiments realistic, we adopt reasonable criteria to generate synthetic datasets of users preferences. Along this line, we avoid synthesising users that always follow trivial rules (e.g., always liking a particular ingredient, and, therefore, any

Table 2
Precision values of the algorithm per user and prescription. Precision values denote actually liked recipes (i.e., true positive) over the all proposed recipes (i.e., true positive plus false positive) obtained by prescriptions and extracted rules.

users	p. 1	p. 2	p. 3	p. 4	p. 5	p. 6	average
user 1	0.831	0.8	0.8621	0.6667	0.6875	0.7857	0.7722
user 2	0.6338	0.2979	0.8	0.7083	0.9268	0.7727	0.6899
user 3	0.7625	0.4333	0.7297	0.75	0.6111	0.6604	0.6578
user 4	0.75	0.6	0.8387	0.65	0.5435	0.7755	0.693
user 5	0.8571	0.7667	0.95	0.7083	0.8182	0.8095	0.8183
user 6	0.6	0.5116	0.8636	0.8261	0.55	0.619	0.6617
user 7	0.7625	0.4667	0.1852	0.8276	0.7391	0.7826	0.6273
user 8	0.85	0.75	0.9048	0.5758	0.6667	0.7872	0.7558
user 9	0.6316	0.9	0.8929	0.8085	0.8936	0.6809	0.8012
user 10	0.875	0.66	0.7692	0.7879	0.717	0.7692	0.763
user 11	0.8625	0.84	0.963	0.7391	0.64	0.8113	0.8093
user 12	0.7875	0.8333	0.9024	0.8108	0.7037	0.8776	0.8192

Table 3
Precision values of the algorithm per user and prescription. Note that precision values denote actually liked recipes (i.e., true positive) over the all proposed recipes (i.e., true positive plus false positive) obtained by prescriptions and NN.

users	p. 1	p. 2	p. 3	p. 4	p. 5	p. 6	average
user 1	0.8267	0.8333	1.0	0.9474	0.8696	0.8913	0.8947
user 2	0.95	0.9474	0.931	1.0	0.95	0.94	0.9531
user 3	0.7875	0.7797	0.7273	0.8478	0.88	0.8163	0.8064
user 4	0.775	0.7833	0.871	0.8158	0.7442	0.7447	0.789
user 5	0.9265	0.9333	0.9565	0.9474	0.9524	0.9375	0.9423
user 6	0.9375	0.9833	1.0	0.9615	0.9592	0.9583	0.9666
user 7	0.8	0.8667	0.9	0.8571	0.8723	0.8913	0.8646
user 8	0.8625	0.7833	0.8837	0.7	0.8077	0.8065	0.8073
user 9	0.7875	0.7833	0.9118	0.8095	0.6786	0.6842	0.7758
user 10	0.8125	0.5667	0.8049	0.7632	0.7959	0.84	0.7639
user 11	0.8875	0.7667	0.7674	0.8723	0.7917	0.7931	0.8131
user 12	0.825	0.8667	0.8049	0.68	0.8246	0.8364	0.8063

recipes containing it). To do so, we introduce some noise in the datasets' synthesis process (e.g., ingredients have preference values inside distributions, classes are stochastically assigned). The overall effect is two folded. Firstly, we discourage situations where users' preferences can be trivially described by logic rules. Secondly, we mimic real-life scenarios where the combination of disparate ingredients in different recipes may affect users' preferences in complex ways.

Table 1 reports the accuracy of the neural networks trained to predict users' preferences, other than the accuracy and fidelity of the logic rules extracted from them. Individual network accuracies range from 0.74 to 0.97 with a mean value of almost 0.86. This is quite a wide variability range. We speculate that the reason why there is a wide gap is due to the aforementioned noise. Each user is different from each others and the difference in their tastes can be huge – like in the real life – therefore some users have more predictable preferences than others. The accuracies of the extracted range from 0.68 to 0.86 with a mean value of 0.76. We notice that there is almost a mean difference of 0.095 between the accuracy of the networks and the accuracy computed on the rules. This fact should not be surprising, given that, in the extraction process, we limited decision trees' depth considerably, therefore the performance of the resulting rules is inherently affected. Similar considerations can be made upon precision measure. Finally, we remind that extracted rules simply *approximate* the internal decision-making process of the NN they have been extracted from. So, their performance cannot overcome the original network's one.

Table 2 summarises the precision obtained during the recommendation phase using prescriptions and the extracted rules of the previous step. We can notice that the precision can change a lot depending on the user and on the prescription. If we look at the mean precision value of all experiments (the last cell of the table), that is about 0.74, we notice that it is quite close to the average precision of rules in Table 1. More in detail, if we apply the Student's *T*-test to the precision values in Table 1 ("r. prec." column) and the precision values in Table 2 ("average" column) we obtain a P-value of 0.386. This means that there is no statistical difference between the two distributions. In other words, the process of recommending recipes that users will like from the whole dataset of available recipes has the same statistical effectiveness of recommending liked recipes prescribed by human experts.

Table 3 summarises the precision obtained during the recommendation phase using prescriptions and the predictions of the sub-symbolic predictors. If we apply the same analysis done to Table 2 we obtain similar results. The P-value of the statistic test is 0.403 bringing us to the same conclusion: the recommendation process has the same effectiveness in proposing prescribed liked recipes and just liked recipes with NN.

5. Discussion

Summarising, experiments show that using sub-symbolic predictors for this personalised food recommendation task is better than using interpretable symbolic predictors—as far as *only* the overall precision is considered. However, the goal of our framework is not to reach higher performance w.r.t. sub-symbolic predictors or other existing systems. Rather, our goal is to help users and experts – i.e. *humans* – to understand *why* recipes are welcome or not. The extracted rules address this issue. In fact, they can be used by experts to adjust their prescription to make them fitter for the user. Moreover, the overall performance obtained by

rules, despite being lower than the one with the NN, it is still quite acceptable in a real-case scenarios.

For example, while trying to motivate any recommendation provided to *user 1* when considering **p.1**, we can easily figure out why the system provided that particular recommendation by looking at which logic rule the system exploited to draw its recommendation. Consider, for instance, the recipe "Shakkara (Sweet) Pongal" (recipe id 4055). It is actually liked by the user 1, and it is composed of: basmati rice, butter, camphor, cardamom, cashew nut, lentils, milk, raisin and sugar. This recipe is recommended because it contains milk and sugar—as we can figure out by looking at the rules extracted for user 1. Dually, if we analyse some random recipe which is *not* liked by the user – such as "Lasagna Spinach Roll-Ups" (recipe id 10,815), made of several ingredients that we do not report for readability — we find out that it is disliked because it contains eggs and pepper.

So, the value added of our symbolic approach lays in the explainability of the whole system—here intended as the possibility to derive motivations for recommendations.

6. Conclusions

In this study we apply symbolic knowledge extraction to design agent-based nutritional recommendation systems. We do so following the purpose of understanding whether it is possible to design nutritional RS which are simultaneously correct (w.r.t. experts prescriptions), acceptable (w.r.t. users' preferences), and *explainable*. Along this line, we propose to adopt logic solvers – instead of bare sub-symbolic predictors – as the underlying engine to generate personalised recommendations. Provided that users preferences can be brought into symbolic form – and it can, via SKE –, we show how the adoption of a symbolic, logic-based way to represent/manipulate knowledge is fundamental to support the explainability feature of nutritional RS—at the price of a negligible reduction of predictive performance. In particular, logic makes it possible to observe how a recommender agent has reached a particular conclusion—and this, in turns, makes it possible to provide motivations to the users.

Future works

As future work, we plan to extend and exploit this work in several directions.

As far as extensions are concerned, we plan to refine our approach so as to support not only recipes/prescription involving ingredients, but also recipes/prescriptions containing *given quantities* of ingredients. This would require slight modifications to the logical modelling of both recipes and prescription, yet it may involve more complexity in the way we train the sub-symbolic predictors, as well as in the way we extract rules out of them. It is worth mentioning that this kind of extension would not affect the overall methodology presented in this study, but it would contribute in making explainable nutritional recommendation systems more realistic.

As far as exploitations are concerned, we plan to integrate the proposed approach into an explainable nutritional recommender system that we can test through user experiments. Moreover, some cultural and economical aspects could be taken into consideration. A person who lives in, say, Asia may have different nutritional preferences from a person who lives in Europe—and vice versa. Similarly, the cost of ingredients may play a key role in user's choices. In the context of the Expectation project (cf. Section 6), which is supporting this work, the possibility of doing such sorts of personalisation is a key outcome. Along this way, our contribution plays a pivotal role in the adaptation of nutritional recommendation systems to disparate user preferences, as well as to the provisioning of motivations for the recommendations.

Statements of ethical approval

This study has been performed involving no patient, no volunteer, and no animal. No personal or sensitive data has been exploited.

Fundings

This work has been partially supported by the Chist-Era IV project "Expectation", the Italian Ministry for Universities and Research (G.A. CHIST-ERA-19-XAI-005), and by the Scientific and Research Council of Turkey (TÜBİTAK, G.A. 120N680).

Declaration of Competing Interest

Authors declare that they have no conflict of interest.

CRediT authorship contribution statement

Matteo Magnini: Conceptualization, Data curation, Formal analysis, Methodology, Validation, Writing – original draft, Writing – review & editing. **Giovanni Ciatto:** Conceptualization, Data curation, Formal analysis, Methodology, Validation, Writing – original draft, Writing – review & editing. **Furkan Cantürk:** Conceptualization, Data curation, Formal analysis, Methodology, Validation, Writing – original draft, Writing – review & editing. **Reyhan Aydoğan:** Conceptualization, Data curation, Formal analysis, Methodology, Validation, Writing – original draft, Writing – review & editing. **Andrea Omicini:** Conceptualization, Data curation, Formal analysis, Methodology, Validation, Writing – original draft, Writing – review & editing.

Acknowledgements

The author would like to thank the anonymous reviewers for their valuable comments and suggestions which lead to significant improvements of this paper.

Appendix A. Details about experiments

A1. Synthesis of users preferences data

The dataset `04_Recipe-Ingredients_Aliases` describes the composition of recipes in terms of ingredients. For each user that we plan to use in our experiments, we build a synthetic dataset starting from the aforementioned one. In particular, we generate a synthetic label for each recipe, stating whether the user likes it or not. Class "like" is encoded as 1, whereas class "dislike" is encoded as 0.

The whole process of assigning a class to a recipe, for user i, is summarised below:

1. a profile is defined for the user i; the profile is a set of value ranges for a number of ingredients/categories
2. for each range, we generate a value with uniform distribution – i.e. a random number guaranteed to lay in the $[-1, 1]$ interval – representing how much the ith user likes that ingredient; if the range targets a category, then a value is generated for each ingredient belonging to that category
3. ingredients without a prior distribution have value equals to 0 (neutral)
4. each recipe r is then assigned with a likeness value ($Ł_r^i$), corresponding to the sum of the likeness values of the ingredients the recipe is made of; so, recipes for which $Ł_r^i \geq 0$ are generally appreciated by user i, whereas recipes with $Ł_r^i < 0$ are not
5. all recipes are ordered according to their likeness value, in descending order

6. recipes in the first quantile are then assigned to class 1 (“like”), while all remaining recipes are either assigned with class 0 (“dislike”) if their Ł_r^i is negative, or randomically to 1 (resp. 0) with probability Ł_r^i (resp. 1 - Ł_r^i)

The rationale behind this choice of class assignment is that each user has its own preferences for some ingredients. Usually, if a recipe is mostly composed by “liked” ingredients, then it is likely appreciated as a whole (recipes in the first quantile). In case of recipes composed by both liked and disliked ingredients in similar proportions, the user may or may not like them (random selection). Finally, recipes that have more disliked ingredients or a few but very much disliked are always rejected by the user.

A2. Computing recommendations efficiently

The recommendation phase of our experiments aims to identify and propose the recipes that are compliant to a given prescription and, simultaneously, liked by the user. To do so we could use a logic engine, such as Prolog [2], but unfortunately current available implementations of logic engines tend to be slow when dealing with tens of thousand of queries. Therefore, because the rules we rely upon are very simple, we exploit the functionalities of the Pandas library [69] to perform the recommendation task in a much faster way.

After the identification of all recommendable recipes we decide to order them by the number of ingredients in ascending way. Then we select the top R recipes with $R = 20$ and the system recommends them. There are two main reasons for this choice: firstly because a user will not likely read all the possible recommendable recipes (they can be hundreds or thousands), secondly because we prefer to recommend recipes more similar to the prescription, i.e., low amount of ingredients (ideally just the ones elicited in the prescription).

References

[1] L. Breiman, J.H. Friedman, R.A. Olshen, C.J. Stone, Classification and Regression Trees, Wadsworth, 1984.

[2] P. Körner, M. Beuschel, J. Barbosa, V.S. Costa, V. Dahl, M.V. Hermenegildo, J.F. Morales, J. Wielemaker, D. Diaz, S. Abreu, G. Ciatto, Fifty years of Prolog and beyond, Theory Pract. Logic Program. 22 (6) (2022) 776–858, doi:10.1017/S1471068422000102.

[3] V. Espín, M.V. Hurtado, M. Noguera, Nutrition for elder care: a nutritional semantic recommender system for the elderly, Expert Syst. 33 (2) (2016) 201–210, doi:10.1111/exsy.12143.

[4] S. Norouzi, A.K. Ghalibaf, S. Sistani, V. Banazadeh, F. Keykhaei, P. Zareishargh, F. Amiri, M. Nematy, K. Etminani, A mobile application for managing diabetic patients' nutrition: a food recommender system, Arch. Iran. Med. 21 (10) (2018) 466. https://pubmed.ncbi.nlm.nih.gov/30415555/

[5] D. Lawo, T. Neifer, M. Esau, G. Stevens, Buying the ‘right’ thing: designing food recommender systems with critical consumers, in: Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems, in: CHI '21, Association for Computing Machinery, New York, NY, USA, 2021, pp. 85:1–13, doi:10.1145/3411764.3445264.

[6] S.-E. Kuo, H.-S. Lai, J.-M. Hsu, Y.-C. Yu, D.-Z. Zheng, T.-W. Hou, A clinical nutritional information system with personalized nutrition assessment, Comput. Methods Programs Biomed. 155 (2018) 209–216, doi:10.1016/j.cmpb.2017.10.029.

[7] H.W. Loh, C.P. Ooi, S. Seoni, P.D. Barua, F. Molinari, U.R. Acharya, Application of explainable artificial intelligence for healthcare: a systematic review of the last decade (2011–2022), Comput. Methods Programs Biomed. 226 (2022) 107161, doi:10.1016/j.cmpb.2022.107161.

[8] D. Calvaresi, S. Eggenschwiler, J.-P. Calbimonte, G. Manzo, M. Schumacher, A personalized agent-based chatbot for nutritional coaching, in: IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology, in: WI-IAT '21, Association for Computing Machinery, New York, NY, USA, 2021, pp. 682–687, doi:10.1145/3486622.3493992.

[9] F. Sabbatini, G. Ciatto, R. Calegari, A. Omicini, Symbolic knowledge extraction from opaque ML predictors in PSyKE: platform design & experiments, Intell. Artif. 16 (1) (2022) 27–48, doi:10.3233/IA-210120.

[10] F. Sabbatini, G. Ciatto, R. Calegari, A. Omicini, Hypercube-based methods for symbolic knowledge extraction: towards a unified model, in: A. Ferrando, V. Mascardi (Eds.), WOA 2022 – 23rd Workshop “From Objects to Agents”, CEUR Workshop Proceedings, vol.3261, Sun SITE Central Europe, RWTH Aachen University, 2022, pp. 48–60. http://ceur-ws.org/Vol-3261/paper4.pdf

[11] T. Cioara, I. Anghel, I. Salomie, L. Barakat, S. Miles, D.P. Reidlinger, A. Taweel, C. Dobre, F. Pop, Expert system for nutrition care process of older adults, Future Gener. Comput. Syst. 80 (2018) 368–383, doi:10.1016/j.future.2017.05.037.

[12] R. Shandilya, S. Sharma, J. Wong, Mature-food: food recommender system for mandatory feature choices a system for enabling digital health, Int. J. Inf. Manag. Data Insights 2 (2) (2022) 100090, doi:10.1016/j.jjimei.2022.100090.

[13] I. Orue-Saiz, M. Kazarez, A. Mendez-Zorrilla, Systematic review of nutritional recommendation systems, Appl. Sci. 11 (24) (2021), doi:10.3390/app112412069.

[14] N. Hezarjaribi, S. Mazrouee, S. Hemati, N.S. Chaytor, M. Perrigue, H. Ghasemzadeh, Human-in-the-loop learning for personalized diet monitoring from unstructured mobile data, ACM Trans. Interact. Intell. Syst. 9 (4) (2019), doi:10.1145/3319370.

[15] G. Agapito, M. Simeoni, B. Calabrese, I. Caré, T. Lamprinoudi, P.H. Guzzi, A. Pujia, G. Fuiano, M. Cannataro, Dietos: a dietary recommender system for chronic diseases monitoring and management, Comput. Methods Programs Biomed. 153 (2018) 93–104, doi:10.1016/j.cmpb.2017.10.014.

[16] R.B. Fraser, S.Z. Turney, An expert system for the nutritional management of the critically ill, Comput. Methods Programs Biomed. 33 (3) (1990) 175–180, doi:10.1016/0169-2607(90)90040-G.

[17] A. Roither, M. Kurz, E. Sonnleitner, The chef's choice: system for allergen and style classification in recipes, Appl. Sci. 12 (5) (2022) 2590.

[18] T.N. Trang Tran, M. Atas, A. Felfernig, M. Stettinger, An overview of recommender systems in the healthy food domain, J. Intell. Inf. Syst. 50 (3) (2018) 501–526, doi:10.1007/s10844-017-0469-0.

[19] W. Wang, L.-Y. Duan, H. Jiang, P. Jing, X. Song, L. Nie, Market2dish: health-aware food recommendation, ACM Trans. Multimed. Comput., Commun., Appl. (TOMM) 17 (2021) 1–19.

[20] W. Min, S. Jiang, R. Jain, Food recommendation: framework, existing solutions, and challenges, IEEE Trans. Multimed. 22 (10) (2020) 2659–2671, doi:10.1109/TMM.2019.2958761.

[21] P. Forbes, M. Zhu, Content-boosted matrix factorization for recommender systems: experiments with recipe recommendation, in: Proceedings of the Fifth ACM Conference on Recommender Systems, in: RecSys '11, Association for Computing Machinery, New York, NY, USA, 2011, pp. 261–264, doi:10.1145/2043932.2043979.

[22] D. Bianchini, V. De Antonellis, N. De Franceschi, M. Melchiori, Prefer: a prescription-based food recommender system, Comput. Stand. Interfaces 54 (2017) 64–75, doi:10.1016/j.csi.2016.10.010. SI: New modeling in Big Data

[23] J. Freyne, S. Berkovsky, Intelligent food planning: personalized recipe recommendation, in: IUI '10: Proceedings of the 15th International Conference on Intelligent User Interfaces, in: IUI '10, ACM, 2010, pp. 321–324, doi:10.1145/1719970.1720021.

[24] Y. Chen, A. Subburathinam, C.-H. Chen, M.J. Zaki, Personalized food recommendation as constrained question answering over a large-scale food knowledge graph, in: Proceedings of the 14th ACM International Conference on Web Search and Data Mining, in: WSDM '21, Association for Computing Machinery, New York, NY, USA, 2021, pp. 544–552, doi:10.1145/3437963.3441816.

[25] Y. Tian, C. Zhang, Z. Guo, Y. Ma, R. Metoyer, N. Chawla, Recipe2vec: multimodal recipe representation learning with graph neural networks, in: Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, 2022, pp. 3448–3454, doi:10.24963/ijcai.2022/479.

[26] C. Trattner, D. Elsweiler, Investigating the healthiness of internet-sourced recipes: Implications for meal planning and recommender systems, in: Proceedings of the 26th International Conference on World Wide Web, in: WWW '17, International World Wide Web Conferences Steering Committee, 2017, pp. 489–498, doi:10.1145/3038912.3052573.

[27] C. Trattner, D. Elsweiler, Food recommender systems: important contributions, challenges and future research directions, ArXiv abs/1711.02760(2017b).

[28] F. Pecune, L. Callebert, S. Marsella, A recommender system for healthy and personalized recipes recommendations, in: A. Sai, H. Schäfer, H. Torkamaan, C. Trattner (Eds.), HealthRecSys 2020: Health Recommender Systems 2020, CEUR Workshop Proceedings, vol. 2684, Sun SITE Central Europe, RWTH Aachen University, 2020, pp. 15–20. http://ceur-ws.org/Vol-2684/3-paginated.pdf

[29] D. Elsweiler, C. Trattner, M. Harvey, Exploiting food choice biases for healthier recipe recommendation, in: SIGIR '17: Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval, ACM, New York, NY, USA, 2017, pp. 575–584, doi:10.1145/3077136.3080826.

[30] A.D. Starke, M.C. Willemsen, C. Trattner, Nudging healthy choices in food search through visual attractiveness, Front. Artif. Intell. 4 (2021), doi:10.3389/frai.2021.621743.

[31] M. Nilashi, D. Jannach, O. bin Ibrahim, M.D. Esfahani, H. Ahmadi, Recommendation quality, transparency, and website quality for trust-building in recommendation agents, Electron. Commer. Res. Appl. 19 (2016) 70–84, doi:10.1016/j.elerap.2016.09.003.

[32] S. Anjomshoae, A. Najjar, D. Calvaresi, K. Främling, Explainable agents and robots: results from a systematic literature review, in: Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems, in: AAMAS '19, International Foundation for Autonomous Agents and Multiagent Systems, 2019, pp. 1078–1088. https://dl.acm.org/doi/10.5555/3306127.3331806

[33] I. Padhiar, O.W. Seneviratne, S. Chari, D. Gruen, D.L. McGuinness, Semantic modeling for food recommendation explanations, in: 2021 IEEE 37th International Conference on Data Engineering Workshops (ICDEW), 2021, pp. 13–19.

[34] R. Yera, A.A. Alzahrani, L. Martínez, Exploring post-hoc agnostic models for explainable cooking recipe recommendations, Knowledge-Based Syst. 251 (C) (2022), doi:10.1016/j.knosys.2022.109216.

[35] D. Calvaresi, G. Ciatto, A. Najjar, R. Aydoğan, L. Van der Torre, A. Omicini, M.I. Schumacher, Expectation: personalized explainable artificial intelligence for decentralized agents with heterogeneous knowledge, in: D. Calvaresi, A. Najjar, M. Winikoff, K. Främling (Eds.), Explainable and Transparent AI and Multi-Agent Systems. Third International Workshop, EXTRAAMAS 2021, Virtual Event, May 3–7, 2021, Revised Selected Papers, Lecture Notes in Computer Science, vol. 12688, Springer Nature, Basel, Switzerland, 2021, pp. 331–343, doi:10.1007/978-3-030-82017-6_20.
[36] S.J. Russell, P. Norvig, Artificial Intelligence: A Modern Approach, fourth ed., Pearson Education, 2021. https://www.pearson.com/en-us/subject-catalog/p/artificial-intelligence-a-modern-approach/P200000003500/9780137505135
[37] W. Sun, D. Niraula, I. El Naqa, R.K. Ten Haken, I.D. Dinov, K. Cuneo, J.J. Jin, Precision radiotherapy via information integration of expert human knowledge and ai recommendation to optimize clinical decision making, Comput. Methods Programs Biomed. 221 (2022) 106927, doi:10.1016/j.cmpb.2022.106927.
[38] G. Ciatto, M. Castiglio, R. Calegari, Logic programming library for machine learning: API design and prototype, in: R. Calegari, G. Ciatto, A. Omicini (Eds.), CILC 2022 – Italian Conference on Computational Logic, CEUR, vol.3204, CEUR-WS, 2022, pp. 104–118. http://ceur-ws.org/Vol-3204/paper_12.pdf
[39] R. Calegari, G. Ciatto, A. Omicini, On the integration of symbolic and sub-symbolic techniques for XAI: a survey, Intell. Artif. 14 (1) (2020) 7–32, doi:10.3233/IA-190036.
[40] R. Calegari, G. Ciatto, V. Mascardi, A. Omicini, Logic-based technologies for multi-agent systems: asystematic literature review, Auton. Agents Multi-Agent Syst. 35 (1) (2021) 1:1–1:67, doi:10.1007/s10458-020-09478-3. Collection "Current Trends in Research on Software Agents and Agent-Based Software Development"
[41] E.M. Kenny, C. Ford, M. Quinn, M.T. Keane, Explaining black-box classifiers using post-hoc explanations-by-example: the effect of explanations and error-rates in XAI user studies, Artif. Intell. 294 (2021) 103459, doi:10.1016/j.artint.2021.103459.
[42] T.-L.B. Tseng, C.-C. Huang, K. Fraser, H.-W. Ting, Rough set based rule induction in decision making using credible classification and preference from medical application perspective, Comput. Methods Programs Biomed. 127 (2016) 273–289, doi:10.1016/j.cmpb.2015.12.015.
[43] L. Franco, J.L. Subirats, I. Molina, E. Alba, J.M. Jerez, Early breast cancer prognosis prediction and rule extraction using a new constructive neural network algorithm, in: Computational and Ambient Intelligence (IWANN 2007), in: LNCS, vol. 4507, Springer, 2007, pp. 1004–1011, doi:10.1007/978-3-540-73007-1_121.
[44] Y. Hayashi, R. Setiono, K. Yoshida, A comparison between two neural network rule extraction techniques for the diagnosis of hepatobiliary disorders, Artif. Intell. Med. 20 (3) (2000) 205–216, doi:10.1016/s0933-3657(00)00064-6.
[45] G. Bologna, C. Pellegrini, Three medical examples in neural network rule extraction, Phys. Med. 13 (1997) 183–187. https://archive-ouverte.unige.ch/unige:121360
[46] B. Baesens, R. Setiono, C. Mues, J. Vanthienen, Using neural network rule extraction and decision tables for credit-risk evaluation, Manag. Sci. 49 (3) (2003) 312–329, doi:10.1287/mnsc.49.3.312.12739.
[47] B. Baesens, R. Setiono, V. De Lille, S. Viaene, J. Vanthienen, Building credit-risk evaluation expert systems using neural network rule extraction and decision tables, in: V.C. Storey, S. Sarkar, J.I. DeGross (Eds.), ICIS 2001 Proceedings, Association for Information Systems, 2001, pp. 159–168. http://aisel.aisnet.org/icis2001/20
[48] M.T.A. Steiner, P.J. Steiner Neto, N.Y. Soma, T. Shimizu, J.C. Nievola, Using neural network rule extraction for credit-risk evaluation, Int. J. Comput. Sci. Netw. Secur. 6 (5A) (2006) 6–16. http://paper.ijcsns.org/07_book/200605/200605A02.pdf
[49] R. Setiono, B. Baesens, C. Mues, Rule extraction from minimal neural networks for credit card screening, Int. J. Neural Syst. 21 (04) (2011) 265–276, doi:10.1142/S0129065711002821.
[50] A. Hofmann, C. Schmitz, B. Sick, Rule extraction from neural networks for intrusion detection in computer networks, in: 2003 IEEE International Conference on Systems, Man and Cybernetics, vol. 2, IEEE, 2003, pp. 1259–1265, doi:10.1109/ICSMC.2003.1244584.
[51] A. Azcarraga, M.D. Liu, R. Setiono, Keyword extraction using backpropagation neural networks and rule extraction, in: The 2012 International Joint Conference on Neural Networks (IJCNN 2012), IEEE, 2012, pp. 1–7, doi:10.1109/IJCNN.2012.6252618.
[52] R.M. Smullyan, First-Order Logic, Ergebnisse der Mathematik und ihrer Grenzgebiete, vol. 43, 2. Folge, Springer, 1968, doi:10.1007/978-3-642-86718-7.
[53] H.J. Levesque, R.J. Brachman, Expressiveness and tractability in knowledge representation and reasoning, Comput. Intell. 3 (1987) 78–93, doi:10.1111/j.1467-8640.1987.tb00176.x.
[54] R.J. Brachman, H.J. Levesque, The tradeoff between expressiveness and tractability, in: Knowledge Representation and Reasoning, in: The Morgan Kaufmann Series in Artificial Intelligence, Morgan Kaufmann, San Francisco, 2004, pp. 327–348, doi:10.1016/B978-155860932-7/50101-1.
[55] J.A. Robinson, A machine-oriented logic based on the resolution principle, J. ACM 12 (1) (1965) 23–41, doi:10.1145/321250.321253.
[56] K.L. Clark, Negation as failure, in: H. Gallaire, J. Minker (Eds.), Logic and Data Bases, Springer, Boston, MA, 1978, pp. 293–322, doi:10.1007/978-1-4684-3384-5_11.
[57] K.R. Apt, The logic programming paradigm and Prolog, in: J.C. Mitchell (Ed.), Concepts in Programming Languages, Cambridge University Press, 2001, pp. 475–508, doi:10.1017/CBO9780511804175.016.
[58] G. Ciatto, R. Calegari, A. Omicini, 2P-Kt: a logic-based ecosystem for symbolic AI, SoftwareX 16 (2021) 100817:1–7, doi:10.1016/j.softx.2021.100817.
[59] M. McCloskey, N.J. Cohen, Catastrophic interference in connectionist networks: the sequential learning problem, Psychol. Learn. Motiv. 24 (1989) 109–165, doi:10.1016/S0079-7421(08)60536-8.
[60] G. Graffieti, G. Borghi, D. Maltoni, Continual learning in real-life applications, IEEE Robot. Autom. Lett. 7 (3) (2022) 6195–6202, doi:10.1109/LRA.2022.3167736.
[61] R. Calegari, G. Ciatto, J. Dellaluce, A. Omicini, Interpretable narrative explanation for ML predictors with LP: a case study for XAI, in: F. Bergenti, S. Monica (Eds.), WOA 2019 – 20th Workshop "From Objects to Agents", CEUR Workshop Proceedings, vol.2404, Sun SITE Central Europe, RWTH Aachen University, 2019, pp. 105–112. http://ceur-ws.org/Vol-2404/paper16.pdf
[62] P.D. Turney, Technical note: bias and the quantification of stability, Mach. Learn. 20 (1995) 23–33.
[63] K. Dwyer, R. Holte, Decision tree instability and active learning, in: J.N. Kok, J. Koronacki, R.L. de Mantaras, S. Matwin, D. Mladenič, A. Skowron (Eds.), Machine Learning: ECML 2007, Springer Berlin Heidelberg, Berlin, Heidelberg, 2007, pp. 128–139.
[64] R.-H. Li, G.G. Belford, Instability of decision tree classification algorithms, in: Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, in: KDD '02, Association for Computing Machinery, New York, NY, USA, 2002, pp. 570–575, doi:10.1145/775047.775131.
[65] Z. Mirzamomen, M.R. Kangavari, A framework to induce more stable decision trees for pattern classification, Pattern Anal. Appl. 20 (2017) 991–1004.
[66] F. Sabbatini, R. Calegari, Clustering-based approaches for symbolic knowledge extraction, XLoKR 2022 - Third Workshop on Explainable Logic-Based Knowledge Representation, Haifa, Israel, 2022, doi:10.48550/arXiv.2211.00234.
[67] F. Sabbatini, R. Calegari, Symbolic knowledge extraction from opaque machine learning predictors: GridREx & PEDRO, in: G. Kern-Isberner, G. Lakemeyer, T. Meyer (Eds.), 19th International Conference on Principles of Knowledge Representation and Reasoning (KR 2022), IJCAI Organization, Haifa, Israel, 2022, pp. 554–563, doi:10.24963/kr.2022/57.
[68] F. Sabbatini, G. Ciatto, A. Omicini, GridEx: an algorithm for knowledge extraction from black-box regressors, in: D. Calvaresi, A. Najjar, M. Winikoff, K. Främling (Eds.), Explainable and Transparent AI and Multi-Agent Systems. Third International Workshop, EXTRAAMAS 2021, Virtual Event, May 3–7, 2021, Revised Selected Papers, Lecture Notes in Computer Science, vol. 12688, Springer Nature, Cham, Switzerland, 2021, pp. 18–38, doi:10.1007/978-3-030-82017-6_2.
[69] W. McKinney, Data structures for statistical computing in Python, in: S. van der Walt, J. Millman (Eds.), Proceedings of the 9th Python in Science Conference, 2010, pp. 56–61, doi:10.25080/Majora-92bf1922-00a.