Argumentation-based Explainable Recommender System with ARES

Riccardo Felici[1], Emanuele De Angelis[2], Alessio Ferrato[1], Maurizio Proietti[2], Giuseppe Sansonetti[1] and Francesca Toni[3]

[1]Roma Tre University, Rome, Italy
[2]CNR-IASI, Rome, Italy
[3]Imperial, London, UK

Abstract

Traditional recommender systems lack transparency, limiting user trust. This paper presents ARgumentation-based Explainable recommender System - ARES, which offers traceable recommendations with explicit reasoning paths. For explainability ARES relies upon ABALearn, a system that learns Assumption-Based Argumentation (ABA) frameworks from positive and negative examples, given a background knowledge. Argumentative explanations are reformulated into natural language via a Large Language Model, linked in ABA logic to prevent hallucinations. The system uses an iterative learning mechanism, guided by ABALearn, and facilitated by an interactive chatbot, to dynamically adapt user profiles.

Keywords

Explainable Recommender Systems, Assumption-based Argumentation, Traceable and Iterative Learning

1. Introduction

Recommender Systems (RSs) are widely used and highly effective tools for guiding users through vast amounts of information and products, offering personalized suggestions across various domains. However, RSs are often *black boxes*, as they provide suggestions failing to explain why, limiting the trust of the users who do not understand the reasons behind the suggestion [1]. This work seeks to address this critical transparency gap by developing an Explainable Recommender System (XRS) [2], built upon *Assumption-based Argumentation* (ABA) frameworks [3, 4].

We propose *ARES (ARgumentation-based Explainable recommender System)*, whose core contribution is using ABA frameworks to provide traceable recommendations, in which every step of the reasoning process, including rules and assumptions leading to a recommendation, is fully explicit and verifiable. This intrinsic traceability supports a complete reconstruction of the entire reasoning process, from the initial background knowledge to the final recommendation, ensuring a very high degree of transparency. The argumentative explanations are then reformulated in natural language using an advanced linguistic model, making it more understandable to the user. Unlike many LLM-based XRSs that might generate plausible but unfaithful explanations, ARES ensures that natural language generation is directly based on the rigorous logic of ABA, significantly reducing the presence of *hallucinations* [5] and guaranteeing explanations aligned with the actual formal reasoning.

ARES features an iterative learning mechanism, which allows the user profiles to be dynamically updated based on preferences and feedback, making suggestions increasingly accurate. The learning process is driven by ABALearn [6, 7], an automated logic-based learning system designed to infer ABA frameworks from positive and negative examples, and background knowledge. The interactive chatbot enables the learnt ABA frameworks to evolve continuously, integrating new user feedback and

3rd International Workshop on Argumentation for eXplainable AI (ArgXAI@ECAI)

riccardofelici7@gmail.com (R. Felici); emanuele.deangelis@iasi.cnr.it (E. De Angelis); alessio.ferrato@uniroma3.it (A. Ferrato); maurizio.proietti@iasi.cnr.it (M. Proietti); gsansone@dia.uniroma3.it (G. Sansonetti); f.toni@imperial.ac.uk (F. Toni)
0000-0002-7319-8439 (E. De Angelis); 0000-0001-5405-3018 (A. Ferrato); 0000-0003-3835-4931 (M. Proietti); 0000-0003-4953-1390 (G. Sansonetti); 0000-0001-8194-1459 (F. Toni)

preferences without requiring a full retraining from scratch. This represents a clear advantage over traditional one-shot learning approaches, which lack such flexibility.

We validate our method in the complex and subjective field of perfumery, showing that ARES can deliver recommendations that are transparent, personalized, and capable of adapting over time, while having comparable or competitive performances against standard baselines.

Related work In the context of RS, various approaches have been developed to enhance explainability and user trust. Xian et al. [8] introduced CAFE, a CoArse-to-FinE neural symbolic reasoning framework that generates user profiles as coarse sketches of behaviors, guiding a path-finding process to derive reasoning paths for recommendations as fine-grained predictions. This method emphasizes the importance of incorporating symbolic reasoning into RS to improve interpretability. Similarly, Tan et al. [9] proposed CountER, a counterfactual explainable recommendation model that utilizes counterfactual reasoning from causal inference to generate minimal changes on item aspects, creating a counterfactual item where the recommendation decision is reversed. This approach aids in providing clear explanations by highlighting what would need to change for a different recommendation outcome.

Several argumentation-based RS have been proposed in the literature to date. Among these, Rago et al. [10] uses quantitative tripolar argumentation frameworks, rather than ABA frameworks, generated automatically from data without any need for knowledge to be manually incorporated, and Rago et al. [11] draw recommendations for a variety of products from their textual reviews, but with quantitative bipolar argumentation rather than ABA, and without integrating user profiles. Furthermore, Briguez et al. [12] use a further form of argumentation (Defeasible Logic Programming) to formulate the conditions under which a movie should be recommended to a given user, but without any learning from examples/background knowledge. To the best of our knowledge, the proposed methodology is the first to use learnt ABA frameworks towards explainable recommendations.

Paper Structure In Section 2 we present background on ABA frameworks and ABALearn. In Section 3 we present our argumentative approach used for the development and implementation of the ARES recommender system. In Section 4 we describe the implementation of iterative learning with ABALearn via a chatbot interface. In Section 5 we present the results of the experimental evaluation, before concluding and discussing future work in Section 6.

2. Background

2.1. Assumption-Based Argumentation Frameworks

An ABA framework [3, 4] is a tuple $\langle \mathcal{L}, \mathcal{R}, \mathcal{A}, ^- \rangle$, where:

- $\mathcal{L}$ is a set of sentences;
- $\mathcal{R}$ is a set of inference rules of the form $s_0 \leftarrow s_1, \ldots, s_n$, with $s_i \in \mathcal{L}$, for $i = 0, \ldots, n$;
- $\mathcal{A} \subseteq \mathcal{L}$ is a non-empty set of assumptions;
- $^- : \mathcal{A} \to \mathcal{L}$ is a contrary function, mapping each assumption to its contrary in $\mathcal{L}$.

In a rule $s_0 \leftarrow s_1, \ldots, s_n$, the sentence s_0 is the *head* of the rule and $s_1, \ldots, s_n$ is the *body* of the rule. If the body of a rule is empty we call it a *fact*. In this work, we focus on *flat* ABA frameworks, where assumption in heads of rules are disallowed. In general, elements of $\mathcal{L}$ can be any sentences, but in this paper we restrict to ABA frameworks where $\mathcal{L}$ is a finite set of ground atoms. However, in the spirit of logic programming, we use schemata to write sentences, rules, assumptions and contraries, using variables that range over a given universe of constants.

Example 1 (Dream Fragrance). To illustrate the fundamental concepts of ABA, let us consider a simple scenario, aimed at determining the liking of a perfume. We define an ABA framework $F = \langle \mathcal{L}, \mathcal{R}, \mathcal{A}, ^- \rangle$ as follows, where X and Y range over a suitable universe to describe perfumes:

- $\mathcal{L} = \{like(X), ingredient\_of(X, Y), floral(Y), \alpha(Y), c\_\alpha(Y)\}$;
- $\mathcal{R} = \{like(X) \leftarrow ingredient\_of(X, Y), floral(Y), \alpha(Y), \quad c\_\alpha(jasmine) \leftarrow,$

$ingredient\_of(dream\_fragrance, rose) \leftarrow$, $ingredient\_of(printemps, jasmine) \leftarrow$, $floral(rose) \leftarrow$, $floral(jasmine) \leftarrow\}$;

- $\mathcal{A} = \{\alpha(Y)\}$;
- $\overline{\alpha(Y)} = c\_\alpha(Y)$.

Intuitively, a perfume X is liked if it contains a floral ingredient Y, unless Y is $jasmine$, and a particular fragrance ($dream\_fragrance$) contains $rose$ and $jasmine$ ingredients, both floral.

We often write facts as rules with equalities in the body, e.g., in the earlier example, we may write $floral(rose) \leftarrow$ as $floral(X) \leftarrow X = rose$.

Given an ABA framework, an *argument* for a *claim* $s \in \mathcal{L}$ is a deduction of s constructed from a finite set of assumptions $A \subseteq \mathcal{A}$ by applications of rules in $\mathcal{R}$ [4]. A constitutes the *support* for the argument.

The acceptability of arguments depends on their ability to defend from possible "attacks": an argument $Arg1$ *attacks* an argument $Arg2$ if the claim of $Arg1$ is the contrary of an assumption in the support of $Arg2$. In this paper, the notion of acceptability we focus on (for flat ABA frameworks) is given in terms of *stable extensions* [3, 4], which determine accepted (and rejected) arguments and their associated claims as follows. A set $\Delta \subseteq \mathcal{A}$ of arguments is a stable extension iff (i) no argument in Δ attacks any argument in Δ (i.e. Δ is *conflict-free*) and (ii) every argument not in Δ is attacked by an argument in Δ (i.e. Δ "attacks" all arguments it does not contain, thus pre-emptively "defending" itself against attacks). We say that an ABA framework is *satisfiable* if it admits at least one stable extension, and *unsatisfiable* otherwise. We also say that a sentence is *(credulously) accepted* in a stable extension Δ of an ABA framework if it is the claim of an argument in Δ.

Example 2 (Dream Fragrance, Cont.). Given F in Example 1, there is an argument $A1$ for claim $like(dream\_fragrance)$ with support $\{\alpha(rose)\}$, and another argument $A2$ for the same claim with support $\{\alpha(jasmine)\}$. $A1$ is not attacked by any other argument of F. In contrast, $A2$ is attacked by the argument $A3$ for $c\_\alpha(jasmine)$ with support the empty set of assumptions. $A1$ and $A3$ belong to the unique stable extension of F (and thus $like(dream\_fragrance)$ is accepted), and $A2$ does not.

An ABA argument can be represented and visualized as an *argument tree* in a "hierarchical" manner. An argument tree is a finite tree whose root node corresponds to the claim, the internal nodes represent the atoms derived by applying rules in the intermediate steps, and the leaves are the assumptions in the support of the argument. The structure of these trees is particularly interesting because it allows us to visualize and trace the reasoning paths that the inferential process has taken to deduce the claim. For an argument tree to be considered well-formed, it must be finite and acyclic [13].

2.2. Learning ABA Frameworks

Learning ABA frameworks aims at generating understandable rules for decision-making, thus helping to promote transparency and interpretability, which are crucial aspects for overcoming the black box nature of many traditional machine learning models. In *ABALearn* [6, 14, 15] the learning process takes as input: (1) a background knowledge, in the form of a satisfiable ABA framework $F = \langle \mathcal{L}, \mathcal{R}, \mathcal{A}, \overline{\ } \rangle$, (2) positive examples $\mathcal{E}^+ \subseteq \mathcal{L}$, and (3) negative examples $\mathcal{E}^- \subseteq \mathcal{L}$, and derives an ABA framework $F' = \langle \mathcal{L}', \mathcal{R}', \mathcal{A}', \overline{\ }' \rangle$, with $\mathcal{R} \subseteq \mathcal{R}'$, $\mathcal{A} \subseteq \mathcal{A}'$, $\overline{\ } \subseteq \overline{\ }'$, such that (i) F' admits a stable extension Δ, (ii) all positive examples are accepted in Δ, and (iii) no negative example is accepted in Δ.

ABALearn learns ABA frameworks automatically by making use of *transformation rules*, including: (1) *rote learning*, which, given a positive example $p(a)$, introduces a new rule $p(X) \leftarrow X = a$; (2) *folding*, which, given rules $H \leftarrow B, C$ and $K \leftarrow B$, derives the new rule $H \leftarrow K, C$; (3) *assumption introduction*, which, given rule $H \leftarrow B$, introduces an assumption α, with contrary $\overline{\alpha}$, and derives the new rule $H \leftarrow B, \alpha$; and (4) *fact subsumption*, which deletes any fact of the form $p(a) \leftarrow$ (or $p(X) \leftarrow X = a$) if there is an accepted argument with claim $p(a)$ in the ABA framework $\langle \mathcal{L}, \mathcal{R} \setminus \{p(a) \leftarrow\}, \mathcal{A}, \overline{\ } \rangle$.

The ABALearn algorithm follows an iterative strategy based on four steps:

1. **Generating initial rules**. This step applies *rote learning* to learn facts from positive examples.

2. **Generalising facts**. This step selects a fact obtained by rote learning and applies *fact subsumption*. If the fact is not subsumed, it applies *folding* with the goal of generating a new, more general, rule that makes no explicit references to the constants occurring in the ABA framework.
3. **Introducing new assumptions**. This step applies *assumption introduction* to any rule obtained by step (2) if it supports an argument for a negative example.
4. **Learning facts for contraries**. This step applies *rote learning* to derive facts for the contraries of the new assumptions introduced by step (3).

The ABALearn strategy consists in following the above steps according to the pattern: $(1); (2; 3; 4)^*$.

Example 3 (Learning ABA frameworks). We now show how the first two rules of $\mathcal{R}$ in Example 1 can be learnt from the facts in $\mathcal{R}$. We assume that the examples are: $\mathcal{E}^+ = \{like(dream\_fragrance)\}$ and $\mathcal{E}^- = \{like(printemps)\}$. By rote learning, we get[1]

$\rho_1.\ like(X) \leftarrow X = dream\_fragrance$

By repeatedly folding, we get:

$\rho_2.\ like(X) \leftarrow ingredient\_of(X, Y), floral(Y)$

Now, we have derived an ABA framework with a single stable extension, where the positive example $like(dream\_fragrance)$ is accepted, but also the negative example $\{like(printemps)\}$ is accepted. To avoid the acceptance of the negative example, by assumption introduction, we add the assumption $\alpha(X)$ to the body of rule ρ_2 and, by rote learning, we add the fact $c\_\alpha(X) \leftarrow X = jasmine$ for the contrary of $\alpha(X)$, thereby getting the set $\mathcal{R}$ of rules shown in Example 1.[2]

When considering the stable extension semantics, ABALearn is implemented in *ASP-ABAlearn* (available at https://github.com/ABALearn/aba_asp) using the SWI-Prolog [16] system and the Clingo [17] ASP solver. . The central idea of ASP is to solve a given computational problem by specifying it as a set of rules, called *ASP program*, whose models, called *answer sets*, represent solutions to the computational problem. In particular, by translating an ABA framework into an ASP program, ABALearn can take advantage of the mapping between stable extensions and answer sets, thereby reducing some reasoning tasks required by rote learning and fact subsumption to computing answer sets of the ASP encoding. Indeed, by inspecting the answer sets of the ASP encoding of an ABA framework, we can learn facts to (i) accept positive examples (at step 1), and (ii) attack assumptions to reject negative examples (at step 4), as well as remove facts (at step 2) whose corresponding claims already belong to the stable extension.

Although the above mentioned mapping would allow us to recast our method for learning (flat) ABA frameworks as a method for learning ASP programs, we believe that working with the ABA representation gives us several advantages. First of all, it reflects more naturally the argumentative approach to the learning process, by which the learnt rules can be considered to be defeasible, and hence they can be modified when conflicting conclusions or exceptions arise. This aspect is especially significant when ABA frameworks are learnt incrementally [18]. Furthermore, the adoption of this formalism allows the direct use of tools for learning ABA frameworks that have been recently developed [6].

3. ARES

The ARES architecture in Figure 1a is designed to support an iterative, learning-based process. The system has been developed for providing recommendations in the perfume domain, but its structure is very general, and could be easily adapted to support a very wide range of different domains. It consists of several interconnected modules that manage the different stages of the recommendation process:

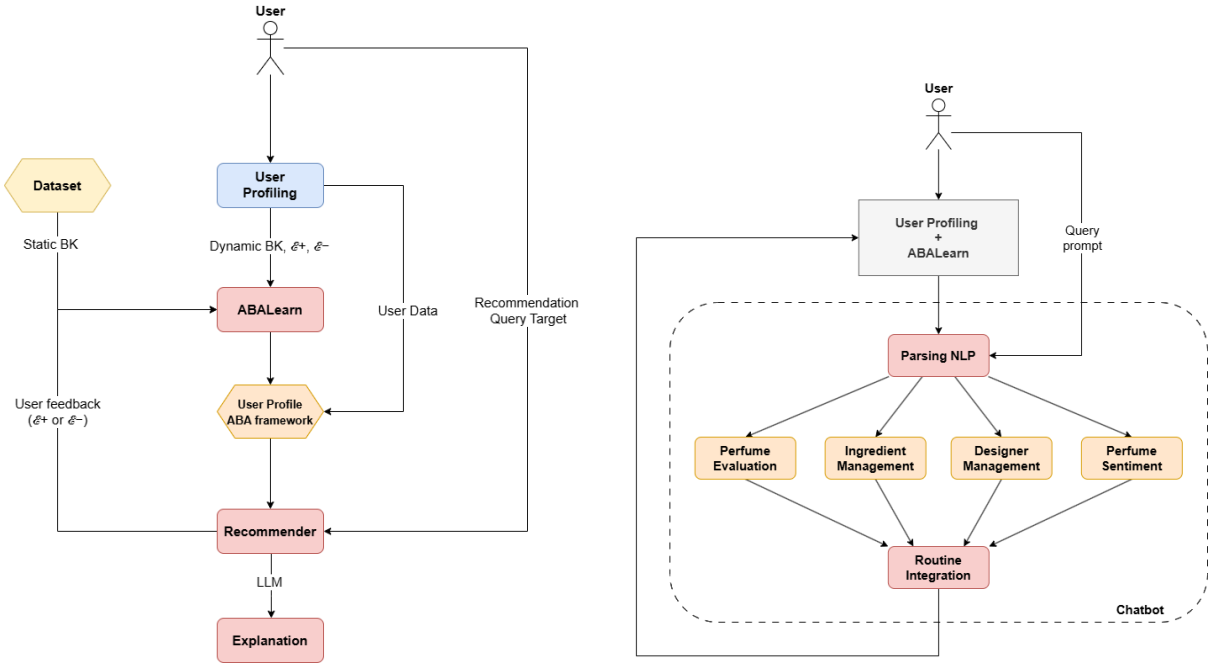
- **User Profiling Module**: Manages the collection of user information and its preparation. It captures explicit and implicit preferences, converting them into positive and negative examples ($\mathcal{E}^+$, $\mathcal{E}^-$), and background knowledge (BK) rules in the ABALearn syntax.

[1] We label rules with identifiers ρ_i for ease of reference.
[2] No applications of fact subsumption have been necessary in this example.

- **ABALearn Module**: Represents the core of ARES. It receives as input the examples ($\mathcal{E}^+$, $\mathcal{E}^-$) and the background knowledge BK. Its task is to learn and continuously update the ABA framework that constitutes the user preference profile.
- **Recommender Module**: Uses the learnt ABA framework to generate personalized recommendations. It also receives specific queries describing the desired features in the perfume sought.
- **Explanation Module**: Generates natural language explanations, using a Large Language Model, and argument trees for the recommendations produced. Input includes recommended item details and logic evidence derived from the answer set via the inferential process.

User Profiling The initial phase of ARES is devoted to acquiring information from the user and constructing her/his personalized profile, represented as an ABA framework. Explicit preferences, such as liked or disliked perfumes indicated directly by the user, are translated into positive and negative examples that will be used to train the ABA framework. Implicit preferences result from the selection of evocative images related to olfactory scenarios or ingredient groups. These selections are used to generate rules that reflect assumptions about the user's preferences. These rules, which model the user's implicit preferences, constitute the ARES' *dynamic* component of the background knowledge, evolving with interaction. For example, if the user likes an image associated with a woody scenario, the system generates a rule stating that the user likes scents containing woody ingredients. To ensure the robustness and flexibility of the learnt rules, *dummy* items may be introduced among the positive examples: the presence of these items prevents the invalidation of the rules in later stages of learning, in the case where these rules are attacked and cover no real examples. Contextually, fragrance domain information, such as ingredients, olfactory scenarios, and designers, are extracted from a dataset and converted to ground facts in the ABA framework, going to constitute the *static* component of the background knowledge, that is, the set of facts describing the domain.

Example 4 (ABA representation of the user profile). We show below an ABA framework representing the profile of a user who likes sweet ingredients. This preference is supported by the first rule and a positive dummy example ($p1$). The rule is defeasible, as it contains an assumption $alpha\_1(A, B)$



(a) Recommender System (b) Chatbot

Figure 1: High-level architectures of the two main modules of ARES: (a) the RS and (b) the chatbot. In Fig. 1b the user profiling and learning modules are a blackbox as their internal structure is unchanged wrt Fig. 1a.

for whose contrary $c\_alpha\_1(A, B)$ ABALearn can learn rules, if needed to capture exceptions. The user's preferences are completed by a positive example (*meltine*) and a negative example (*velvette*). We use the Prolog-like syntax accepted by the ABALearn system with ':-' to indicate '←'. The ABA framework representation also includes declarations of assumptions and their contraries.

```
% Rules from the user profile for dummy example
like(A) :- ingredient_of(A,B), sweet(B), alpha_1(A,B).
sweet(A) :- A=t1.
ingredient_of(A, B) :- A=p1, B=t1.

% Rules from the perfume dataset for non-dummy examples
oriental(A) :- A=vanilla.
ingredient_of(A, B) :- A=meltine, B=vanilla.

sweet(A) :-  A=sugar.
ingredient_of(A, B) :- A=velvette, B=sugar.

% Assumptions and contraries
assumption(alpha_1(A,B)).
contrary(alpha_1(A,B),c_alpha_1(A,B)) :- assumption(alpha_1(A,B)).

% Positive examples: [like(p1),like(meltine)]
% Negative examples: [like(velvette)]
```

Examples and background knowledge are given as input to ABALearn, which produces an ABA framework that represents the user profile, and is capable of supporting arguments for or against liking certain perfumes. In particular, the learnt framework accepts all positive example, does not accept any negative example, and is also able to decide the acceptance of unseen items.

Example 5 (Learnt ABA framework). From the ABA framework, positive and negative examples in Example 4, ABALearn generates a rule for the contrary of the assumption $alpha\_1(A, B)$. This rule captures an exception to the rule representing a preference for sweet scents, and excludes the ingredient *sugar* to avoid accepting the negative example $like(velvette)$. A rule for `like` is also generated to cover the positive example $like(meltine)$, stating the preference for oriental ingredients. The output is:

```
c_alpha_1(A,B) :- ingredient_of(A,B), B=sugar.
like(A) :- ingredient_of(A,B), oriental(B).
```

The learning process is inherently dynamic: when new information or feedback is collected from the user, it is used to update the examples or background knowledge, triggering a re-execution of ABALearn. However, as we will see in detail in Section 4, by rendering learnt rules defeasible, ABALearn is able to modify existing rules and add exceptions without having to rerun the training from scratch. This mechanism allows refining the ABA framework and adapting the user profile dynamically over time.

Recommender The recommendation generation phase uses the learnt ABA framework to identify and suggest (unseen) perfumes that the user may like. The process begins by capturing the user-specified recommendation goal, such as seasonality or desired scent intensity. A preliminary filtering step is then applied to reduce the pool of candidate perfumes. This filter uses the characteristics desired by the user and compares them with the attributes in the dataset to create an initial ranking. This ranking is based on how consistent each perfume is with the user's specified preferences and objective scent attributes; it does not consider the ABA framework user profile. For each candidate perfume that has

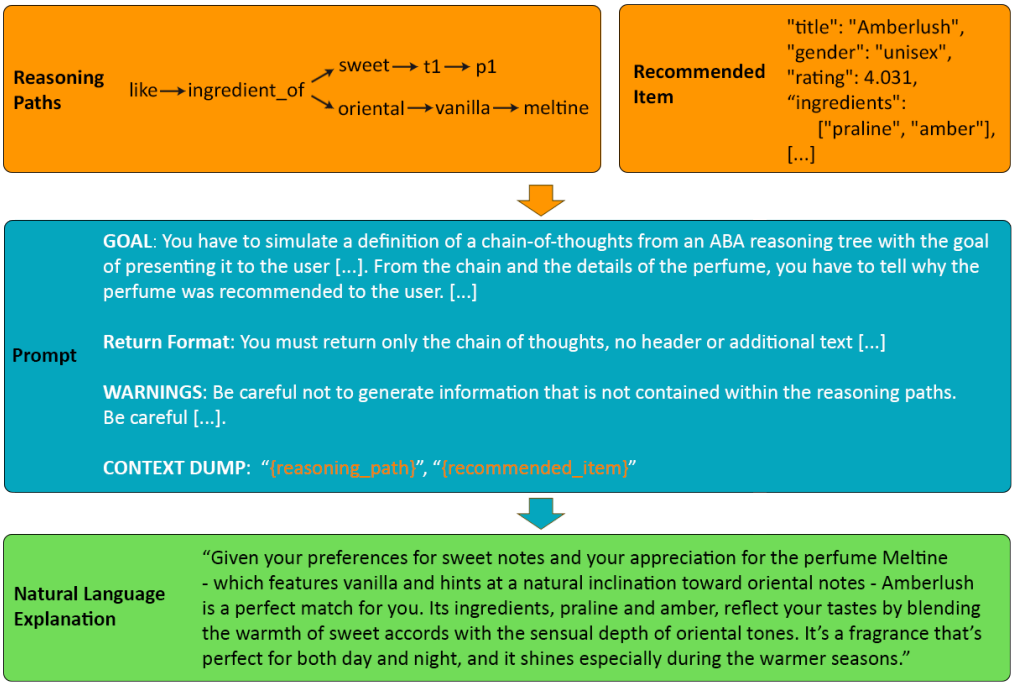


Figure 2: Pipeline for generating the natural language explanation

passed preliminary filtering, its attributes are transformed into ground ABA facts. The user's ABA framework and the candidate perfume facts are translated into an ASP program, and Clingo determines which perfumes specified by the user's goal belong to the answer set of that program, and hence are claims accepted in an ABA framework's stable extension. An analysis of the answer set produced by Clingo makes it possible to identify which specific rules of the ABA framework were used to derive the recommended claims from the facts. The number of used rules helps calculate a similarity score, indicating the degree of compatibility between the perfume and the user profile. This score is combined with the preliminary rank to obtain a final rank, computed for item A as:

$$\text{Ranking}(A) = 0.5 \cdot C(A) + 0.5 \cdot R(A) \tag{1}$$

Where $C(A)$ represents the argumentative contribution, which is quantified by the number of supporting rules derived by Clingo. $R(A)$ incorporates a combination of heuristic parameters relating the relevance of A to the request. These parameters include ratings, matching features, genre compatibility and differences in intensity.

Although the similarity formula presented here has been calibrated specifically for perfumery, the modularity of the ARES architecture allows alternative, more generic similarity metrics to be integrated. This modular approach ensures that the system's high-level structure remains unchanged, enabling ARES to adapt effectively to different scenarios and domains. While the current choice aims at achieving an optimal balance between generality and accuracy for the specific use case, it also paves the way for future extensions to get broader applicability.

Explanation The recommendations generated are not simply suggestions, but are supported by an explicit formal reasoning process, where each step is verifiable. This is made possible by the system's ability to construct argument trees and *reasoning paths*, defined as the specific paths within the argument tree, which link facts in the background knowledge (such as user preferences and item features) to the final recommendation claim. This transparency makes it possible to determine which features or rules

influenced the decision. To translate these reasoning paths into user-understandable explanations, the system makes use of the Gemini 2.0 Flash model. The LLM receives the extracted reasoning paths as input, along with other formal argumentation evidence, and, guided by a carefully formulated prompt, generates a natural language explanation. This prompt is designed to constrain the LLM to produce consistent descriptions that are faithful to the argumentative process, ensuring that the explanation is firmly linked to the actual data and inferences, thus preventing the generation of hallucinations.

This approach differs from some recent trends in Explainable AI, such as *Chain of Thought* (CoT) *prompting* [19]. Although CoT prompting aims to explicate the intermediate steps of a model's reasoning, recent studies [20] have shown that LLMs can generate CoTs that are plausible but not always sound and reflecting the actual process of constructing an answer, thus creating an illusion of reasoning [21]. In our system, however, the faithfulness of the reasoning chain is guaranteed: the reasoning paths are not an arbitrary textual construction, but are derived directly from facts and rules of the ABA framework. Thus, the generated explanation is verifiable and corresponds faithfully to the actual inferential process, offering robust and reliable explainability.

Example 6 (Recommendation, reasoning paths and explanation). Reasoning paths are extracted from a single argument tree generated by the recommendation for the item *amberlush*. With reference to Examples 4 and 5, Figure 2 shows a path that supports the $like$ rule for sweet ingredients and a path for oriental ingredients. The prompt uses the argument tree and the recommended item details to generate a natural language explanation.

4. Iterative Learning

The ARES chatbot architecture in Figure 1b serves as an advanced conversational interface for user interaction. Its main purpose is to facilitate the collection of information and feedback from the user in natural language, overcoming the limitations of traditional structured interfaces. It allows the user to rate and ask questions about perfumes, and provide comments indicating liking or disliking. Information gathered through the chatbot is used to dynamically update the user's profile and refine future recommendations. This architecture is built by emulating an *agentic* style [22], where the system adapts its behavior based on the user's request.

- **NLP Parsing**: Receives user textual prompts as input and uses *Natural Language Processing* (NLP) techniques to parse the text, identify expressed communicative intent, and extract relevant entities (item name, class and sentiment). The implementation leverages the capabilities of an LLM to classify user requests [23].
- **Operational Routines**: Based on the output of the NLP Parsing Module, the message is routed to one of the specialized operational routines. Each routine is designed to perform specific operations in response to particular categories of user requests.
- **Routine Integration Module**: This module represents the point of convergence of the various operational routines. It is responsible for aggregating and standardizing the outputs generated by the individual specialized routines, ensuring a consistent, processable format for integration at later stages of the learning cycle. The new structured information is then passed on to the User Profiling and ABALearn modules.

The interaction through the chatbot enables the iterative learning of the ABA framework representing the user's profile. Information and feedback acquired through the conversational interface is formalized in the format needed by the ABALearn engine, with feedback management routines playing a key role in this formalization. When the user expresses a positive sentiment toward a specific feature, the system generates a new $like$ rule within the user's ABA framework. Similarly to the case of the user profiling module, dummy items are introduced to serve as explicit positive examples. The handling of negative sentiment, on the other hand, is more complex and is addressed through an integrated approach with

ABALearn: information about disliked items is formalized through the creation of a dummy item with the negative feature, which is added to the set of negative examples provided in input to ABALearn. This mechanism allows the learning engine to identify existing rules that enforce liking of the dummy item and automatically generate the necessary contrary to prevent such derivation. Finally, feedback related to specific perfumes results in a direct update of the lists of positive and negative examples in the user profile, allowing the user to change her/his mind about previously expressed preferences.

Once all the collected and formalized information has updated the inputs, the learning process is re-executed. This iterative cycle, unlike static models, allows the ABA framework to evolve dynamically, incorporating new knowledge from more recent user interactions, with the goal of progressively improving the accuracy and relevance of future recommendations. This feature of ARES can be seen as a realisation of *contestable learning* [24, 18], which gives users the ability to interact with the system and question its decisions or recommendations. In other words, users can question the rules that lead to the acceptance of an undesired claim and, vice versa allow the system to learn new rules that lead to the acceptance of a desired claim. The *redress* of the system after contestation is obtained by making the previously learnt rules defeasible, and then learning new rules, without re-learning from scratch.

Example 7 (Contestation: Ingredient with negative sentiment). After the *amberlush* recommendation received in Example 6, the user may contest the system by marking *amber* as an undesirable feature. To redress the ABA framework after this contestation, a new dummy item $p2$ is created, together with rules specifying that $p2$ has the *amber* ingredient and that *amber* is an *oriental* type of scent. Then, ABALearn identifies any rule that can be used for entailing $like(p2)$. This is the previously learnt rule $like(A) \leftarrow ingredient\_of(A, B), oriental(B)$ (see Example 5). Now, by applying the assumption introduction transformation, ABALearn adds a new assumption $alpha\_2(X)$, thus rendering the rule defeasible and, by rote learning, also adds a rule for the contrary $c\_alpha\_2(X)$ of the assumption, thus introducing an exception to the general rule:

```
% Background knowledge for dummy item p2
ingredient_of(A, B) :- A=p2, B=amber.
oriental(B) :- B=amber.

% Learnt rules
like(A) :- ingredient_of(A,B), oriental(B), alpha_2(A,B).
c_alpha_2(A, B) :- ingredient_of(A, B), B=amber.
```

5. Evaluation

The ARES evaluation process was conducted to quantify the effectiveness and performance of the proposed model by comparing it with established methodologies. The evaluation methodology employed standard quantitative metrics and a cross-validation protocol based on the *Leave-One-Out Cross-Validation* technique [25]. This technique involves temporarily removing a single item from each user's profile for use as a test item, training the system on the reduced profile and evaluating its ability to correctly predict the omitted item. For comparison, several *Collaborative Filtering* algorithms were used, including *KNN (User-based Nearest-Neighbor)*, *SVD (Singular Value Decomposition)*, *NMF (Non-negative Matrix Factorization)* and *CoClustering*.

The evaluation metrics adopted include *Mean Absolute Error* (MAE) and *Root Mean Square Error* (RMSE), which measure the accuracy of numerical predictions, and *Precision@n*, *Recall@n*, and *F-Measure@n*, which assess the quality of recommendations in terms of relevance and completeness.

The experimental results in Table 1 show that the ARES approach performs in line with State-of-Art algorithms in terms of MAE and RMSE, indicating good predictive accuracy. However, a lower recall was observed for ARES than for the collaborative algorithms, suggesting that in highly subjective domains such as perfumery, collective preferences may offer added value over purely content analysis. Overall, our experiments show that our approach does not sacrifice too much performance with respect

Table 1
Comparative evaluation. We report in bold the best values for each metric. (∗) Time per user. (†) Tests run on VMware virtual machine with Fedora Linux 42, 20 GB RAM, AMD Ryzen™ 5 3600, NVIDIA RTX 3050.

Metric	KNN	SVD	NMF	CoClustering	ARES
MAE	**0.9225**	1.0740	1.0001	1.0360	0.9649
RMSE	1.3574	1.2495	1.4504	1.4970	**1.2351**
Precision@5	0.1623	0.1629	0.1621	**0.1644**	0.0952
Recall@5	0.7982	0.8017	0.7974	**0.8077**	0.4216
F-measure@5	0.2697	0.2707	0.2694	**0.2732**	0.1657
Precision@10	0.0811	0.0814	0.0811	**0.0822**	0.0558
Recall@10	0.7982	0.8017	0.7974	**0.8077**	0.4216
F-measure@10	0.1473	0.1479	0.1472	**0.1492**	0.1069
Precision@15	0.0541	0.0543	0.0540	**0.0548**	0.0381
Recall@15	0.7982	0.8017	0.7974	**0.8077**	0.4216
F-measure@15	0.1013	0.1017	0.1012	**0.1026**	0.0744
Training Time (s)†	80.37	0.53	1.44	1.63	9.16*

to State-of-the-Art systems, while providing transparency and explainability. A comparison with other explainable RS, e.g., CAFE [8] and CountER [9], is left for future work.

Besides explainability, a distinctive feature of ARES is its support to contestability (see Section 4). The effects of contestability are evaluated through an iterative simulation that compared ARES with the KNN algorithm. This simulation allowed us to observe the trend of the mean absolute error over several iterations for a single user, with five simulations. The graph in Figure 3 associated with this test revealed that ARES converges to a lower absolute error, demonstrating remarkable adaptive ability and consistency in the representation of preferences. Alternatively, the KNN model showed less stable behavior, with the absolute error fluctuating along iterations without clear convergence. This result emphasizes how our approach is particularly suitable in scenarios where user preferences evolve over time, thus confirming its validity from a dynamic perspective and its ability to adapt effectively.

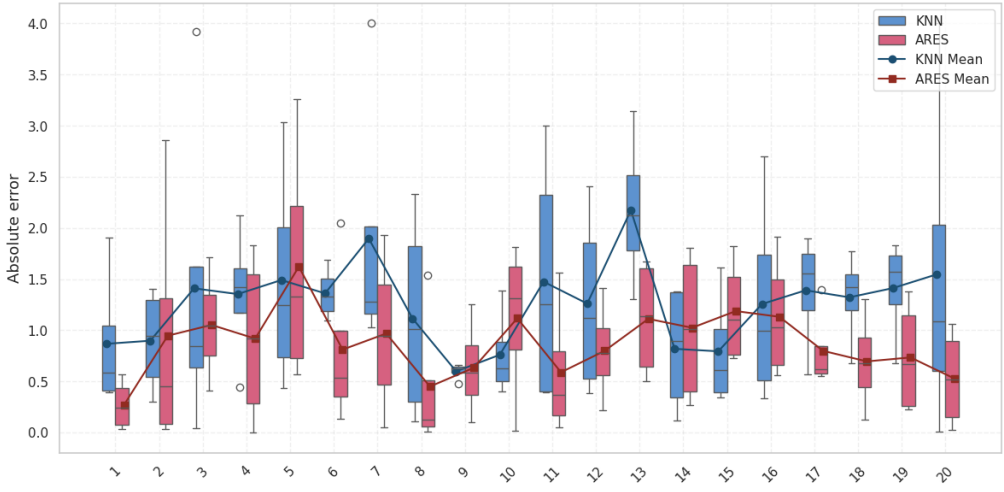


Figure 3: Absolute error calculated on an average of 5 simulations of 20 iterations for a single user.

6. Conclusions

We explored the application of ABALearn for the development of explainable recommender systems, addressing the problem of opacity inherent in traditional models. We introduced ARES (ARgumentation-

based Explainable recommendation System), a system that generates traceable and understandable recommendations based on an explicit reasoning process. Our architecture integrates a formal representation via ABA frameworks, which model the user's preference profile through rules, assumptions and contraries, and an LLM to translate argumentative explanations into natural language. The linking of explanations to formal derivations in ABA ensures the faithfulness of the inferential process and prevents the generation of hallucinations.

A distinguished feature of ARES is its iterative and adaptive learning mechanism, enabled by a conversational chatbot interface. This dynamic interaction enables the system to continuously update the user profile based on explicit and implicit feedback, allowing the ABA framework to evolve and refine recommendations over time.

Future Work. We envisage several directions for the future development of ARES. A first direction concerns the implementation of a hybrid model that combines the ARES content-based approach with collaborative filtering. Instead of considering only the interaction history, we could exploit the similarity between users, based on the similarity between the ABA frameworks generated for each of them, e.g., by comparing extensions (or answer sets of the corresponding ASP representations). This would enable us to take advantage of the personalisation inherent in content-based user profiles and the ability of collaborative filtering to detect common patterns among users with similar logical preference structures.

A second line of development focuses on presenting logical reasoning as Chain of Thoughts in natural language. Currently, explanations are generated by the LLM from the extracted reasoning paths. We could further explore automatic textual generation of these reasoning paths, making them even more understandable and narrative for the user. The ultimate goal is to provide a chain of reasoning that is not only plausible, but whose soundness and faithfulness to the inference process is ensured by direct derivation from the rules of the ABA framework, unlike approaches generating an illusion of reasoning.

Finally, ARES focuses on the perfumery domain, but its design principles and architecture are general. We plan to implement other instances of the system in different domains. It is reasonable to expect that in domains where the features of the items to be recommended are more objective (e.g., technological or financial products), the benefits of ARES may be even more pronounced.

Acknowledgments

We thank support by the Royal Society, UK (IEC\R2\222045). Toni was funded by the ERC (grant agreement No. 101020934) and by J.P. Morgan and the RAEng, UK, under the Research Chairs Fellowships scheme (RCSRF2021\11\45). De Angelis and Proietti were supported by the MUR PRIN 2022 Project DOMAIN funded by the EU-NextGenerationEU (2022TSYYKJ, CUP B53D23013220006, PNRR, M4.C2.1.1), the PNRR MUR project PE0000013-FAIR (CUP B53C22003630006), and the INdAM - GNCS Project *Argomentazione Computazionale per apprendimento automatico e modellazione di sistemi intelligenti* (CUP E53C24001950001). De Angelis and Proietti are members of the INdAM-GNCS research group.

Declaration on Generative AI

The authors have not employed any Generative AI tools.

References

[1] A. Said, On explaining recommendations with large language models: a review, Frontiers in Big Data 7 (2025). doi:10.3389/fdata.2024.1505284.

[2] Y. Zhang, X. Chen, Explainable recommendation: A survey and new perspectives, Foundations and Trends® in Information Retrieval 14 (2020) 1–101. doi:10.1561/1500000066.

[3] A. Bondarenko, P. Dung, R. Kowalski, F. Toni, An abstract, argumentation-theoretic approach to default reasoning, Artificial Intelligence 93 (1997) 63–101.

[4] P. Dung, R. Kowalski, F. Toni, Assumption-Based Argumentation, 2009, pp. 199–218. doi:10.1007/978-0-387-98197-0_10.
[5] Z. Xu, S. Jain, M. Kankanhalli, Hallucination is inevitable: An innate limitation of large language models, 2025. arXiv:2401.11817.
[6] E. De Angelis, M. Proietti, F. Toni, Learning brave assumption-based argumentation frameworks via ASP, in: Proceedings of ECAI 2024, volume 392 of *FAIA*, IOS Press, 2024, pp. 3445–3452. doi:10.3233/FAIA240896.
[7] C. Tirsi, M. Proietti, F. Toni, ABALearn: An automated logic-based learning system for ABA frameworks, in: Proceedings AIxIA 2023, LNCS 14318, Springer, 2023, pp. 3–16. doi:10.1007/978-3-031-47546-7_1.
[8] Y. Xian, Z. Fu, H. Zhao, Y. Ge, X. Chen, Q. Huang, S. Geng, Z. Qin, G. de Melo, S. Muthukrishnan, Y. Zhang, CAFE: Coarse-to-Fine Neural Symbolic Reasoning for Explainable Recommendation, in: Proceedings CIKM 2020, Association for Computing Machinery, New York, NY, USA, 2020, p. 1645–1654. doi:10.1145/3340531.3412038.
[9] J. Tan, S. Xu, Y. Ge, Y. Li, X. Chen, Y. Zhang, Counterfactual explainable recommendation, in: Proceedings CIKM 2021, Association for Computing Machinery, New York, NY, USA, 2021, p. 1784–1793. doi:10.1145/3459637.3482420.
[10] A. Rago, O. Cocarascu, F. Toni, Argumentation-based recommendations: Fantastic explanations and how to find them, in: Proceedings IJCAI 2018, ijcai.org, 2018, pp. 1949–1955. doi:10.24963/IJCAI.2018/269.
[11] A. Rago, O. Cocarascu, J. Oksanen, F. Toni, Argumentative review aggregation and dialogical explanations, Artif. Intell. 340 (2025) 104291. doi:10.1016/J.ARTINT.2025.104291.
[12] C. E. Briguez, M. C. Budán, C. A. D. Deagustini, A. G. Maguitman, M. Capobianco, G. R. Simari, Argument-based mixed recommenders and their application to movie suggestion, Exp. Sys. with Applications 41 (2014) 6467–6482. doi:10.1016/j.eswa.2014.03.046.
[13] R. Craven, F. Toni, Argument graphs and assumption-based argumentation, Artificial Intelligence 233 (2016) 1–59. doi:https://doi.org/10.1016/j.artint.2015.12.004.
[14] E. De Angelis, M. Proietti, F. Toni, Greedy ABA learning for case-based reasoning, in: Proceedings AAMAS 2025, IFAAMAS, 2025, p. 556–564.
[15] E. De Angelis, M. Proietti, F. Toni, ABA learning via ASP, in: Proceedings ICLP 2023, volume 385 of *EPTCS*, 2023, pp. 1–8. doi:10.4204/EPTCS.385.1.
[16] J. Wielemaker, T. Schrijvers, M. Triska, T. Lager, SWI-Prolog, TPLP 12 (2012) 67–96.
[17] M. Gebser, R. Kaminski, B. Kaufmann, T. Schaub, Clingo = ASP + Control: Preliminary report, 2014. arXiv:1405.3694.
[18] E. De Angelis, M. Proietti, F. Toni, Learning to contest argumentative claims, in: Proceedings RuleML+RR 2025, LNCS, Springer, (to appear).
[19] B. Wang, S. Min, X. Deng, J. Shen, Y. Wu, L. Zettlemoyer, H. Sun, Towards understanding chain-of-thought prompting: An empirical study of what matters, 2023. arXiv:2212.10001.
[20] D. Paul, R. West, A. Bosselut, B. Faltings, Making reasoning matter: Measuring and improving faithfulness of chain-of-thought reasoning, 2024. arXiv:2402.13950.
[21] P. Shojaee, I. Mirzadeh, K. Alizadeh, M. Horton, S. Bengio, M. Farajtabar, The illusion of thinking: Understanding the strengths and limitations of reasoning models via the lens of problem complexity, 2025. arXiv:2506.06941.
[22] J. Schneider, Generative to agentic AI: Survey, conceptualization, and challenges, 2025. URL: https://arxiv.org/abs/2504.18875. arXiv:2504.18875.
[23] M. He, P. N. Garner, Can ChatGPT detect intent? Evaluating large language models for spoken language understanding, 2023. arXiv:2305.13512.
[24] F. Leofante, H. Ayoobi, A. Dejl, G. Freedman, D. Gorur, J. Jiang, G. Paulino-Passos, A. Rago, A. Rapberger, F. Russo, X. Yin, D. Zhang, F. Toni, Contestable AI needs computational argumentation, 2024. arXiv:2405.10729.
[25] Z. Meng, R. McCreadie, C. Macdonald, I. Ounis, Exploring data splitting strategies for the evaluation of recommendation models, 2020. arXiv:2007.13237.