Probabilistic rule induction for transparent CBR under uncertainty

Martin Jedwabny, Pierre Bisquert, Madalina Croitoru

Probabilistic rule induction for transparent CBR under uncertainty

Martin Jedwabny,[1] Pierre Bisquert,[1, 2] Madalina Croitoru[1]
[1]LIRMM, Inria, Univ Montpellier, CNRS, Montpellier, France
[2]IATE, INRAE, Institut Agro, Montpellier, France
martin.jedwabny@lirmm.fr, pierre.bisquert@inrae.fr,
madalina.croitoru@lirmm.fr

Abstract

CBR systems leverage past experiences to make decisions. Recently, the AI community has taken an interest in making CBR systems explainable. Logic-based frameworks make answers straightforward to explain. However, they struggle in the face of conflicting information, unlike probabilistic techniques. We show how probabilistic inductive logic programming (PILP) can be applied in CBR systems to make transparent decisions combining logic and probabilities. Then, we demonstrate how our approach can be applied in scenarios presenting uncertainty.

1 Introduction

Case-based reasoning (CBR) [1] is a problem-solving technique that uses previously encountered experiences to solve a problem. When novel problems are encountered, similar past cases are retrieved and their solutions are adapted to the situation at hand. Furthermore, soft case-based reasoning [14] allows for uncertainty in the form of missing information or when a single description has been solved using multiple solutions. Here, we focus on this second type of uncertainty, in which a single problem might have been solved using different solutions in the past. Indeed, this scenario can easily present itself as a consequence of multiple domain experts providing conflicting solutions, or when a single one has different confidence levels over the space of possible solutions.

Lately, there has been a major interest from the Artificial Intelligence community to make systems that provide explanations to their answers, giving birth to what is known as explainable AI (XAI). Due to several concerns stemming from the lack of transparency and interpretability [3], several approaches have been proposed to make up for these issues for various AI subfields, including CBR systems [20]. Indeed, some literature [19, 12] even includes computing explanations as part of the fundamental workflow of CBR systems.

Past literature in XAI has developed two main ways of achieving explainability [3]: transparent systems and post-hoc models to explain black-box systems [16]. Some authors consider the second type of explanations as limited due to the fact that they provide reconstructions which might not be really linked to the actual reasoning process of the black-box system. On the other hand, transparent systems often sacrifice the quality of the answers or computational performance, in order to fully offer interpretable results.

Recently, there has been a growing interest in the XAI community in frameworks that combine symbolic and probabilistic approaches, which allows to both handle uncertainty in the form of conflicting information, and deliver transparent results. A state-of-the-art probabilistic rule induction system, called ProbFOIL+ [7] allows to infer probability annotated rules from noisy data. However, this approach will often suffer from time complexity issues when the available information is too large.

In this paper, we argue that case-based reasoning in conjunction with ProbFOIL+ is a feasible and interesting approach to both tackling the time complexity issues of probabilistic rule induction, and providing transparent answers in soft case-based reasoning problems featuring problems with conflicting solutions.

The following sections are structured as follows. Section 2 introduces the basic notions required to present our framework. Then, section 3 covers the general architecture of our framework, the CBR case representation, and the reasoning mechanism from a theoretical perspective. Section 4 goes into the implementation details of our system. In section 5, the reader will find a the results obtained by the implementation of our framework. Finally, section 6 summarizes our results, discusses different perspectives upon which our work contributes to, and compares our work to previous research.

2 Preliminaries

In this section, we describe the logical language we will use to represent the cases of our CBR system and some basic concepts of probabilistic inductive logic programming [7] which our system uses to perform the case adaptation phase described in the upcoming sections.

2.1 Logic language

We build upon a first-order logic language $\mathcal{L}$ composed of *constants* $\{a, b, \ldots\}$, *variables* $\{X, Y, \ldots\}$, *function* symbols $\{f, g, \ldots\}$, and *predicate* symbols $\{P, Q, \ldots\}$. A *term* t is a constant, a variable or a functor. An *atom* $P(t_1, \ldots, t_n)$ is a predicate P of arity $n \in \mathbb{N}_0$ applied to terms $t_1, \ldots, t_n$. A *rule* $H \leftarrow B_1, \ldots, B_n$ is a construct composed of a *head* atom H and a finite conjunction of *body* atoms $B_1, \ldots, B_n$. A *fact* is a rule with an empty body. A *substitution* $\theta = \{X_1/t_1, \ldots X_k/t_k\}$ is a mapping from variables to terms. Applying a substitution θ to an atom a is denoted as $a\theta$ and it replaces the variables in the domain

of θ with their corresponding terms. In the case $a\theta$ contains no variables, we say that the operation is a grounding of a and we call $a\theta$ a ground atom.

2.2 Probabilistic logic language

We can extend logic rules with probabilistic annotations in order to deal with uncertain information. ProbLog [8] is a probabilistic first-order logic language that extends the notions presented above with *probabilistic rules* (and facts) $p_i :: r_i$ where $p_i \in [0, 1]$ denotes a probability and r_i is a rule. Its semantics is based on what is known as a distribution semantics [18] (a well-known semantics for probabilistic logics).

A ProbLog *program* $T = \{p_1 :: r_1, \ldots, p_n :: r_n\}$ consists of a finite set of probabilistic rules. Given a finite set of possible grounding substitutions $\{\theta_{i,1}, \ldots, \theta_{i,m_i}\}$ for each probabilistic rule $p_i :: r_i \in T$, a ProbLog program T defines a probability distribution over the subsets $L \subseteq L_T$ of possible groundings $L_T = \{r_1\theta_{1,1}, \ldots, r_1\theta_{1,n_1}, \ldots, r_n\theta_{n,1}, \ldots, r_n\theta_{n,m_n}\}$ of (strict) rules in T as:

$$P(L \mid T) = \prod_{r_i\theta_j \in L} p_i \prod_{r_i\theta_j \in L_T \setminus L} (1 - p_i)$$

Moreover, ProbLog defines the *success probability* of a query q (i.e. finite conjunction of atoms) as the overall probability that a random subset $L \subseteq L_T$ entails q:

$$P_s(T \models q) = \sum_{\substack{L \subseteq L_T \\ L \cup D \models q}} P(L \mid T)$$

2.3 Probabilistic inductive logic programming

PILP [7] is a subfield of statistical relation learning (SRL) [10] that addresses the task of inferring a set of logic rules that justify a target predicate from examples and background knowledge, when the examples and learnt rules may be annotated with probabilities.

As mentioned before, the PILP setting is the task of finding an hypothesis in the form of a set of probabilistic rules from which a set of examples of a target predicate can be derived with minimal loss. More precisely (based on [7]):

Definition 1 (PILP problem) *Given the following:*

1. *A set of examples E, composed of pairs (x_i, p_i) where x_i is a grounding for the target predicate t and p_i its probability,*
2. *A background theory B containing information related to the examples in the form of a ProbLog program,*
3. *A loss function $\text{loss}(H, B, E)$, measuring the error of a hypothesis (set of rules) H w.r.t B and E (in [7]: $loss(H, B, E) = \sum_{(x_i,p_i)\in E} |P_s(B \cup H \models x_i) - p_i|$), and*

4. A space of possible clauses L_h specified as in [13].

Find a hypothesis $H \subseteq L_h$ *such that* $H = \underset{H' \subseteq L_h}{\operatorname{argmin}}\ loss(H', B, E)$.

Notice that the definition above accounts for probabilistic rules in the background theory B and hypothesis H unlike [7], which only accounts for probabilistic facts. However, as mentioned in [8] this definition is equivalently general, as probabilistic rules can be replaced in this setting by strict rules by adding a fresh probabilistic fact (with the same probability annotation) to its body. We modified this definition for ease of use in later sections.

3 Our framework

In this section, we will present our framework from a theoretical perspective, while the implementation details will be described in the following one. It utilizes the mechanisms of probabilistic inductive logic programming to adapt previously encountered cases so that it allows them to have conflicting solutions.

As mentioned before, CBR is a methodology which consists of finding and reusing past problems to solve novel ones, which present similar features. We will denote the *case base* (i.e. the past cases) of a CBR system as a collection of (possibly non-unique) pairs $\mathcal{CB} = \{(x_i, y_i)\}_i$ where $x_i \in \mathcal{P}$ is called the problem and $y_i \in \mathcal{S}$ its solution.

Both $\mathcal{P}$ the *problem space* and $\mathcal{S}$ the *solution space*, are typically assigned a fixed structure by the CBR implementation. More precisely, their structure should be contained in its prespecified *background knowledge* $\mathcal{BK}$ consisting of (i) the case base structure, (ii) integrity constraints, (iii) the case base itself, and (iv) additional knowledge for the many stages of the decision process.

Given a case base and a novel problem x^{new}, a CBR system should assign an appropriate solution y^{new}. More precisely, case-based reasoners suppose that $\mathcal{CB}$ follows an unknown relation $Sols \subseteq \mathcal{P} \times \mathcal{S}$ and its objective is to replicate its behaviour such that $(x^{new}, y^{new}) \in Sols$.

When confronted with a novel case x^{new}, CBR systems normally follow an architecture consisting of four distinct steps [1]:

1. *Retrieve* previously seen cases that are similar to the current one. This step is often performed by extracting the k cases from $\mathcal{CB}$ that maximize a similarity function (reflexive and symmetric) $sim : \mathcal{P} \times \mathcal{P} \mapsto [0, 1]$ between the case in question and x^{new}.

2. *Reuse* the retrieved cases from the previous step by selecting or integrating their solutions to produce a set of candidate solutions.

3. *Revise* the candidate solution(s) and adapt it(them) into a solution for x^{new}.

4. *Retain* the new solution after validation into the case base if necessary.

3.1 Representation

We will model our cases using a first-order logical language $\mathcal{L}$ as the one presented in section 2.3. The solution space will be characterized by a set of (reserved) constants $\mathcal{S} = \{s_1, s_2, \ldots\}$. Then, we say that:

Definition 2 (Case) *A case base $\mathcal{CB} = \{(x_i, y_i)\}_i$ is a collection of (possibly non-unique) cases where $y_i \in \mathcal{S}$, and x_i is a set of ground atoms of the form $P(a_1, \ldots, a_n)$ or $P(s, a_1, \ldots, a_n)$ where $s \in \mathcal{S}$, P is a predicate and $a_1, \ldots, a_n$ are (non-solution) constants ($\nexists i, j, a_j \neq s_i$) of $\mathcal{L}$.*

The intuition behind this model is that the atoms of the form $P(a_1, \ldots, a_n)$ describe the scenario of the case, whereas $P(s, a_1, \ldots, a_n)$ are used to represent the properties of each possible solution, denoted by $s \in \mathcal{S}$. This representation will later allow us to encode the adaptation phase of the CBR cycle as a PILP problem as defined in the previous section.

In order to exemplify our framework, let us consider this example from [2]:

Example 1 (Assisted driver) *We set ourselves in the context of an AI-equipped vehicle which can spontaneously take control from the human driver under dangerous circumstances. As such, each case revolves around either taking control of the vehicle, or doing nothing. These two options are always mutually exclusive. The characterization of the problem is given by a set of duties at stake. These are (i) prevention of collision, (ii) respect for driver autonomy, (iii) keeping within speed limit, and (iv) prevention of imminent harm to people.*

More precisely, each case x_i is represented by a set of predicates of the form $Duty(Option, Value)$ where $Duty \in \{preventCollision, respectAutonomy, withinLimit, preventHarm\}$, $Option \in \{takeControl, doNothing\}$, $Value \in \{yes, no\}$, and each solution $y_i \in \{takeControl, doNothing\}$. For example, if a case contains the predicate $preventCollision(takeControl, yes)$, it would imply that taking control of the car in the current scenario guarantees preventing a collision. If a duty is not present in a case, it's deemed as irrelevant.

- *Case 1: There is an object ahead in the driver's lane and the driver moves into another lane that is clear.*

  $x_1 = \{preventCollision(takeControl, yes), respectAutonomy(takeControl, no),$
  $preventCollision(doNothing, yes), respectAutonomy(doNothing, yes)\}$

- *Case 2: The driver has been going in and out of his/her lane with no objects discernible ahead.*

  $x_2 = \{preventCollision(takeControl, yes), respectAutonomy(takeControl, no),$
  $preventCollision(doNothing, yes), respectAutonomy(doNothing, yes)\}$

- *Case 3: The driver is speeding to take a passenger to a hospital. The GPS destination is set for a hospital.*

$$x_3 = \{respectAutonomy(takeControl, no), withinLimit(takeControl, yes), preventHarm(takeControl, no), respectAutonomy(doNothing, yes), withinLimit(doNothing, no), preventHarm(doNothing, yes)\}$$

- *Case 4: Driving alone, there is a bale of hay ahead in the driver's lane. There is a vehicle close behind that will run the driver's vehicle upon sudden braking and he/she can't change lanes, all of which can be determined by the system. The driver starts to brake.*

$$x_4 = \{preventCollision(takeControl, no), respectAutonomy(takeControl, no), preventHarm(takeControl, yes), preventCollision(doNothing, no), respectAutonomy(doNothing, yes), preventHarm(doNothing, no)\}$$

3.2 Reasoning

Having defined our structure for cases, we can now describe our framework using the classical CBR cycle. As explained before, it's composed of four distinct phases, (i) *retrieval* of similar past cases, (ii) *reuse* of their solutions by integration, (iii) *revision* of candidate solutions, and (iv) *retention* of the new solution after validation.

Retrieval

Given a novel case (x, y) defined using our representation above, the reasoner searches for similar encountered cases in its base. Typically, CBR systems have a fixed limit $K \in \mathbb{N}$ of cases to retrieve. The system starts by determining the K past cases that maximize a similarity function $sim : \mathcal{P} \times \mathcal{P} \mapsto [0, 1]$. There are numerous ways of defining such a function in the literature [5]. For our purposes, we'll use a generic yet general weighted function:

$$sim(x_1, x_2) = \frac{\sum\limits_{p \in A(\mathcal{CB})} w_p \times sim_p(x_1, x_2)}{\sum\limits_{p \in A(\mathcal{CB})} w_p} \tag{1}$$

Where $x_1, x_2 \in \mathcal{P}$ are two problems, $A(\mathcal{CB})$ is the set of all ground atoms of $\mathcal{L}$ in $\mathcal{CB}$, $sim_p(x_1, x_2)$ is a local similarity function, and $w_p \in \mathbb{R}_{>0}$ is an optional weight describing the relative importance of p.

For a given case base $\mathcal{CB}$, we denote the K most similar cases to problem $x \in \mathcal{P}$ as $sim^K_{\mathcal{CB}}(x) \subseteq \mathcal{P}$, containing exactly K elements if $|\mathcal{CB}| \geq K$, or all the problems in the base otherwise.

Adaptation

Then, the *reuse* and *revision* steps are implemented by (i) generalizing the solutions from $sim_{\mathcal{CB}}(x^{new})$ through PILP, and (ii) computing the probability that an answer is correct for x^{new} using the ProbLog semantics.

Step (i) reduces generalization to the PILP setting as in Procedure 1. Given a case base $\mathcal{CB}$, an input problem x^{new}, and its K most similar cases $sim^K_{\mathcal{CB}}(x^{new})$, the procedure reduces the adaptation of the these cases to a PILP problem.

Procedure 1: Generalize results from the retrieval step using PILP

Input: Case base $\mathcal{CB}$, problem x^{new}, cases $C = sim^K_{\mathcal{CB}}(x^{new})$
Output: Hypothesis H

1 $E \leftarrow \{(answer(id_x, y), p_y)\}$ such that $(x, y) \in C$ are retrieved cases, 'id_x' maps a problem (set of ground atoms) to a (fresh) constant, '*answer*' is the target predicate of arity 2, and $p_y = \frac{|\{(x,y)\in C\}|}{|\{(x,y')\in C : y\neq y'\}|}$.

2 $B \leftarrow P(id_x, a_1, ..., a_n)$ (or, $P(id_x, s, a_1, ..., a_n)$) for each predicate $P(a_1, ..., a_n)$ (or, $P(s, a_1, ..., a_n)$) in x, for all $(x, y) \in C$.

3 Define $loss(H, B, E) = \sum_{(x_i,p_i)\in E} |P_s(B \cup H \models x_i) - p_i|)$ as in [7].

4 Define the space of possible clauses L_h such that:

- Only allows the variabilized atom $answer(X, Y)$ in a rule's head.
- Only allows atoms of the form $P(X, a_1, \ldots, a_n)$ or $P(X, Y, a_1, \ldots, a_n)$ where $P(a_1, \ldots, a_n)$ or $P(s, a_1, \ldots, a_n)$ are atoms appearing in C, respectively, X, Y are the variables appearing in the rule's head, and $a_1, \ldots, a_n$ remain as constants (are not replaced by variables).

5 Find a hypothesis $H \subseteq L_h$ such that $H = \underset{H'\subseteq L_h}{\text{argmin}}\ loss(H', B, E)$.

The intuition behind this transformation is that it allows conflicting solutions for the same problem by defining p_i as the proportion of times the answer was chosen. In doing so, a PILP solver obtains as output a set of ProbLog rules of the form:

$$p_i :: answer(X, Y) \leftarrow P_{i_1}(X, a_{i_{1,1}}, ..., a_{i_{1,n_1}}), ..., P_{i_m}(X, a_{i_{m,1}}, ..., a_{i_{m,n_m}})$$
$$Q_{i_1}(X, Y, a_{i_{1,1}}, ..., a_{i_{1,n_1}}), ..., Q_{i_t}(X, Y, a_{i_{t,1}}, ..., a_{i_{t,n_t}})$$

Where p_i is the probability annotation of the ProbLog rule, $answer(X, Y)$ the target predicate, P, Q body predicates, X, Y variables, and $a_{i_{j,k}}$ constants.

Example 2 (Assisted driver continued) *Suppose $\mathcal{CB}$ is composed of 1 instance of each pair $(x_1, doNothing)$, $(x_2, takeControl)$, $(x_3, doNothing)$, $(x_4, takeControl)$ and 4 instances of $(x_2, doNothing)$, $(x_3, takeControl)$, $(x_4, doNothing)$. For*

this case base, a hypothesis that minimizes the loss function is the following:

$$0.8 :: answer(A, B) \leftarrow preventHarm(A, B, yes), preventCollision(A, B, no)$$
$$0.8 :: answer(A, B) \leftarrow respectAutonomy(A, B, yes), preventHarm(A, B, yes)$$
$$0.2 :: answer(A, B) \leftarrow withinSpeedLimit(A, B, yes)$$
$$0.2 :: answer(A, B) \leftarrow preventCollision(A, B, yes)$$

Notice that for ProbLog semantics, the last rule obtained by the solver, i.e: $0.2 :: answer(A, B) \leftarrow preventCollision(A, B, yes)$*, doesn't imply there is 0.2 probability of a solution being correct given that it prevents a collision, but rather, that there is a 0.2 chance that the rule itself is valid.*

It's also worth mentioning that the space and time complexity of finding H depends on the PILP implementation. However, our transformation produces a knowledge base of size linear with respect to the amount of ground clauses in C (retrieved cases). This in turn is extremely useful as various sizes K of similarity bounds can be tested at decision-making time. For a experimental analysis of an existing PILP solver please refer to [7]. A more in depth theoretical analysis of the time and space complexity of our framework as a whole is left for future work.

Step (ii) of the adaptation phase consists of executing a ProbLog solver [7] using the hypothesis H obtained from phase (i), (optional) background knowledge B that can be used to enforce certain behaviour at decision-making time, and the ground atoms in the current problem x^{new}. In summary this ProbLog program consists of:

1. H the hypothesis obtained from step (i), i.e: a set of ProbLog rules with head $answer(X, Y)$,
2. B the (optional) background knowledge,
3. All facts of the form $P(id_{x^{new}}, a_1, \ldots, a_n)$ or $P(id_{x^{new}}, s, a_1, \ldots, a_n)$ contained in x^{new}, where $id_{x^{new}}$ is a (fresh) constant, and
4. A query $q = answer(id_{x^{new}}, Y)$ where Y is the variable to be grounded.

Example 3 (Assisted driver continued) *Consider the novel case 5, characterized by:*

$$\begin{aligned} x_5 = \{&respectAutonomy(takeControl, no), preventHarm(takeControl, no),\\ &preventCollision(takeControl, yes), respectAutonomy(doNothing, yes),\\ &preventHarm(doNothing, yes), preventCollision(doNothing, yes)\} \end{aligned}$$

Computing the query $q = answer(x_5, Y)$ *for the hypothesis* H *obtained in the previous part of the example, and adding the ground facts of* x_5*, gives us:*

$$answer(x_5, doNothing) : 0.84$$
$$answer(x_5, takeControl) : 0.2$$

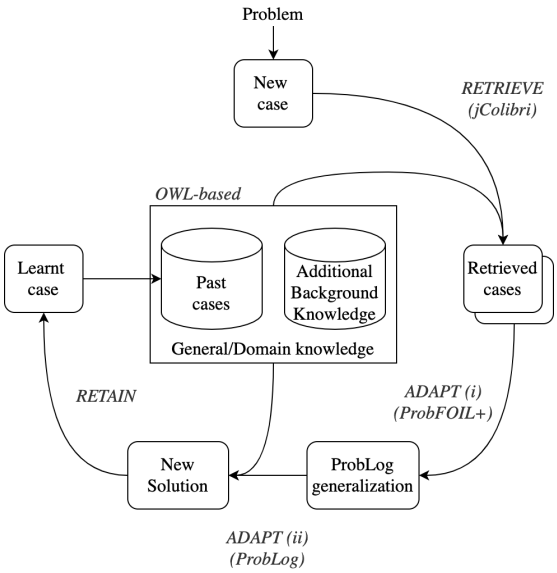

Figure 1: Overview of the architecture of our CBR system

With the result of step (ii), the response of our CBR framework is determined by the solution with the highest probability of being correct according to the ProbLog solver. Notice that the sum of the probabilities don't have to add up to 1, because we want to allow multiple solutions to be valid at the same time, which can be useful depending on the application domain.

Retention

Finally, the last step of the CBR cycle is implemented by adding the case with the infered solution after validation from a human user, if retention is necessary in the application domain's context. Because our main focus was on the adaptation phase, we leave other retention strategies [6] for future work.

4 Implementation

Having defined the overall idea of our framework in the previous section, here we'll describe our current implementation and the technologies involved in it. Figure 1 depicts the high-level architecture of our CBR system.

For the representation of our framework, we used a simplified version of CBROnto [9], an OWL-based domain-agnostic ontology for representing CBR systems. CBROnto allows to define cases, complex properties, similarity measures, and many other useful elements for a CBR system. At the same time, it can combat the main problem of knowledge-intensive CBR systems, the 'knowledge acquisition bottleneck' by reusing knowledge from other ontology

libraries to create complex knowledge structures. This choice of design was useful twofold: (i) it facilitated case and general knowledge elicitation by organizing the information in a generic yet extensible manner, and (ii) it allowed the system to reuse previous research work based on CBROnto.

Our system was mainly developed using the Java language, and is comprised of several subsystems that implement the different phases of the CBR cycle.

The *retrieval* step is performed by using jColibri [15] as a Java library. jColibri is an adaptable CBR framework and full-fledged system that offers an overarching architecture for the phases of case-based reasoning. In our case, we mainly used the library version for the *retrieval* and *retention* steps. It allowed us to easily parse and query the OWL knowledge base in order to retrieve the K most similar cases while being flexible enough to implement our own similarity measures.

As for the similarity-based retrieval function, we used a modified version of the one described in Equation 1. In our version, the local similarity measure used was the *fdeep* function mentioned in [15]. This method utilizes the hierarchical description-logics based similarity of the ontology to determine which properties of two problems are similar. In other words, given a complex hierarchy of types and subtypes in the OWL knowledge base for the properties of a case base problem, this local similarity function will characterize properties as having a higher measure of similarity when they are closer in the hierarchy tree. For example, two unrelated properties will be less similar than two properties of the same type, or when the type of one in a subtype of the other.

Next, the *adaptation* phase was implemented as along the lines of the previous section:

1. After retrieving the K most similar cases from the base, our program translates the problems, solutions and their properties to a ProbLog program and appends a series of directives to specify the hypothesis constraints as in Procedure 1.

2. Then, it runs a version of the ProbFOIL+ algorithm [7] which is publicly available [1] with minimal modifications for efficiency, such as restricting the constants inside predicates to their types, enabling symmetry breaking and a set of generic integrity constraints to prune the search space while preserving generality. Let us stress that the ProbFOIL+ algorithm is an approximate algorithm that searches for the optimal hypothesis with some greedy subroutines and maintaining a bounded approximation of the score of partial answers. As such, we found that these integrity constraints had a considerable impact in the efficiency of the algorithm. For more information, please refer to the implementation link below.

3. Subsequently, our system parses the result of the ProbFOIL+ algorithm and it appends them with the description of the novel case. This in turn generates a ProbLog program that is run with its standard implementation

[1] https://bitbucket.org/problog/prob2foil/.

[2]. The mechanism for transforming this information to a ProbLog program is straightforward, and we only add the queries before execution time.

4. Finally, the result of the ProbLog program is parsed and our system choses the solution for which the success probability as described previously is the highest.

The implementation of our system is publicly available [3].

5 Experimentation

For our experiments, we constructed a dataset from an online game-like survey based on the TV series 'Breaking bad', in which users were presented with various ethically nuanced situations and given a set of possible alternatives to choose from. Users chose a character from TV series and where tasked to give a series of 5-7 answers for the series of questions that guided the overall story leading into one of many possible paths. In total, we acquired over 150 user histories with an average of 6 answers for each user.

Similar to the running example, we represented the problem x in each case (x, y) by the duties each solution fulfills. I.e. as a set of values $Duty(Option, Value)$ where $Duty \in \{fidelity, reparation, gratitude, non\text{-}maleficence, beneficence, self\text{-}improvement, justice\}$, $Option$ a possible solution for the problem, and $Value \in \{extremely\text{-}bad, really\text{-}bad, bad, neutral, good, really\text{-}good, extremely\text{-}good\}$.

These duties were inspired by Ross's [17] theory of *prima facie* duties:

- Fidelity: strive to keep promises and be honest.
- Reparation: make amends when we have wronged someone.
- Gratitude: repay others when they perform actions that benefit us.
- Non-maleficence: refrain from harming others in any way.
- Beneficence: improve other peoples health and well-being.
- Self-improvement: improve our own health and well-being.
- Justice: be fair and try to distribute benefits and burdens evenly.

Each answer provided by the users was stored using our OWL case base architecture as a pair (x, y) with x as described before and y the solution chosen from a predefined set of choices which was described in natural language.

We evaluated our system by in two ways, a quantitative and a qualitative manner. These tests were performed at least 10 times each and the numbers depicted in the figures represent averages over these runs. They were performed in a 1,6 GHz Intel Core i5 computer with 8 GB 2133 MHz LPDDR3 RAM.

[2] https://github.com/ML-KULeuven/problog.
[3] https://github.com/martinjedwabny/cbr-edm

The first experiment was designed to test the computational cost of the proposed mechanisms. In particular, we tested for different choices of K, the added cost of the retrieval and adaptation phases, of which, unsurprisingly the generalization step inside of the adaptation was the most costly. We can see the results in Figure 2. For reference, when $K = 50$ the total amount of facts produced by our translation of this form was around 250, and over 550 in total counting past solution examples and data-type predicates. Using our current system, we can see the adaptation time growing steadily. In practice, the system could handle cases up until $K = 50$ in at most around 2 minutes. However, finer grained experimentation would be required in future work to determine the acceptable bounds of the execution time.

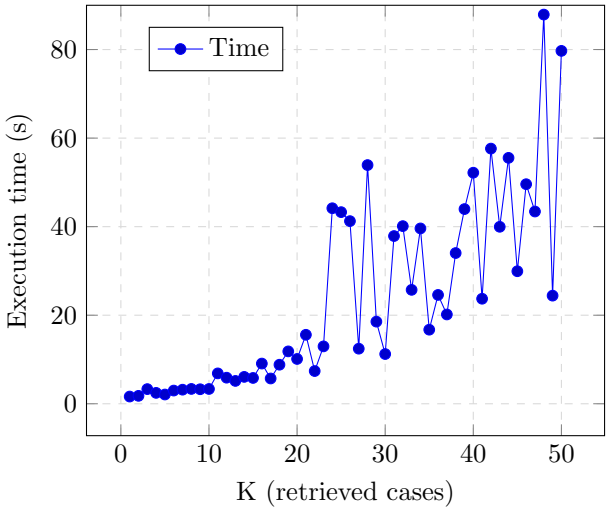

Figure 2: Running time assessment

Then second, we tested the quality of the generalizations made by testing over a reserved dataset containing 16 situations, and inserting the rest in the case base. An example of an answer obtained during these tests is the one shown in Figure 3.

$$0.43877551 :: answer(A, B) \leftarrow hasDuty(A, B, 'justice', 'really\_good').$$
$$0.55555556 :: answer(A, B) \leftarrow hasDuty(A, B, 'beneficence', 'really\_good').$$
$$0.5 :: answer(A, B) \leftarrow hasDuty(A, B, 'honesty', 'good').$$
$$0.2 :: answer(A, B) \leftarrow hasDuty(A, B, 'non\_maleficence', 'really\_good').$$

Figure 3: Example result of the adaptation step

We analysed the quality of the answer using the loss equation defined before:

K	10	20	30	50
loss	9.581	9.874	12.464	10.224

Table 1: Testing loss as the sum of absolute errors

$loss(H, B, E) = \sum_{(x_i,p_i)\in E} |P_s(B \cup H \models x_i) - p_i|)$. Table 1 shows the results obtained. As we can see, the loss is relatively similar for $K = 10$ and $K = 20$ and then degrades heavily. For reference, the predicted probabilities were tested against the 16 situations with 2 possible solutions each, making up for a total of 32 ground atoms to predict, and a maximum loss of 32 (the sum of the absolute values of their differences). The results, while showing room for improvement, seem reasonable enough especially considering the complexity of ethical decision making. With a simple model of ethics as the one presented here, the system could consistently predict reasonable answers in this domain. However, a greater K didn't improve the results, potentially showing that only including the most related cases in our domain can improve the performance of our system, and large K values degraded it by including less related ones. Moreover, it would be interesting to see what results our system would get with a finer characterization of the domain.

6 Conclusion

We have developed a framework for case-based reasoning which can handle noisy datasets and provide explainable solutions to novel cases by profiting from the adaptability of the probabilistic inductive logic programming setting. Then, we described the implementation of our system, which is publicly available, making it easier to specify cases and elicit knowledge by utilizing various state-of-the-art technologies from the representation to the retrieval and adaptation stages of our system. We have depicted how our system fares against a real-world inspired scenario in the context of a ethical dilemmas and shown results from the computational complexity and qualitative viewpoints.

Discussion

As mentioned in [1], CBR differs from machine learning approaches in that it favours learning from experience as a natural by-product of problem solving, as opposed to generalizing from it. Our work adds to this research venue by proposing a further technique with which uncertainty can be handled in CBR systems. Concretely, we combine the ideas of CBR and probabilistic inductive logic programming by generalizing/learning only after the similarity-based retrieval step from CBR systems.

By leveraging the noise-handling capabilities of ProbFOIL+, our system can also be seen as a contribution to soft case-based reasoning [14]. This in turn, shows the adaptability and generality of the probabilistic logic setting, which has

also been previously used to develop explainable AI systems in other subfields, such as decision-making [21] and recommender systems[4].

The manner in which our framework handles uncertainty is also related to multi-criteria fuzzy decision making systems [11] in that it allows multiple decision makers to provide different answers for the same problem, which our system uses to compute a consensus. However, our CBR adds two layers to the reasoning process, namely the similarity-based case selection which filters decisions to the most relevant cases, and secondly the abstraction layer in the form of background knowledge which can be used to extend the properties of cases and potentially use different past cases to compute the answer of a novel one.

From a XAI perspective, our framework is a contribution to *transparent* systems [3], by making it easy for the user to understand the reasoning process, and increasing the confidence in the solution of the system by making the reasoning steps of the CBR cycle depend on well-defined rules.

Future work

We wish to expand our work in the scale of experimentation and compare the quality of our results to other similar CBR systems. It would also be interesting to compare the performance of our reduction of the adaptation step in the CBR cycle to other different encodings. In addition, other datasets coming from both knowledge-intensive and knowledge-light CBR domains could be used to test our implementation.

References

[1] A. Aamodt and E. Plaza. Case-based reasoning: Foundational issues, methodological variations, and system approaches. *AI communications*, 7(1):39–59, 1994.

[2] M. Anderson and S. L. Anderson. Geneth: A general ethical dilemma analyzer. *Paladyn, Journal of Behavioral Robotics*, 9(1):337–357, 2018.

[3] A. B. Arrieta, N. Díaz-Rodríguez, J. Del Ser, A. Bennetot, S. Tabik, A. Barbado, S. García, S. Gil-López, D. Molina, R. Benjamins, et al. Explainable artificial intelligence (xai): Concepts, taxonomies, opportunities and challenges toward responsible ai. *Information Fusion*, 58:82–115, 2020.

[4] R. Catherine and W. Cohen. Personalized recommendations using knowledge graphs: A probabilistic logic programming approach. In *Proceedings of the 10th ACM conference on recommender systems*, pages 325–332, 2016.

[5] P. Cunningham. A taxonomy of similarity mechanisms for case-based reasoning. *IEEE Transactions on Knowledge and Data Engineering*, 21(11):1532–1543, 2008.

[6] R. L. De Mantaras, D. McSherry, D. Bridge, D. Leake, B. Smyth, S. Craw, B. Faltings, M. L. Maher, M. T COX, K. Forbus, et al. Retrieval, reuse, revision and retention in case-based reasoning. *The Knowledge Engineering Review*, 20(3):215–240, 2005.

[7] L. De Raedt, A. Dries, I. Thon, G. Van den Broeck, and M. Verbeke. Inducing probabilistic relational rules from probabilistic examples. In *Proceedings of 24th international joint conference on artificial intelligence (IJCAI)*, volume 2015, pages 1835–1842. IJCAI-INT JOINT CONF ARTIF INTELL, 2015.

[8] L. De Raedt and A. Kimmig. Probabilistic (logic) programming concepts. *Machine Learning*, 100(1):5–47, 2015.

[9] B. Dıaz-Agudo and P. A. González-Calero. Cbronto: a task/method ontology for cbr. *Procs. of the 15th International FLAIRS*, 2:101–106, 2002.

[10] L. Getoor and B. Taskar. Statistical relational learning, 2007.

[11] D. H. Hong and C.-H. Choi. Multicriteria fuzzy decision-making problems based on vague set theory. *Fuzzy sets and systems*, 114(1):103–113, 2000.

[12] J. L. Kolodner. An introduction to case-based reasoning. *Artificial intelligence review*, 6(1):3–34, 1992.

[13] S. Muggleton. Inverse entailment and progol. *New generation computing*, 13(3-4):245–286, 1995.

[14] S. K. Pal and S. C. Shiu. *Foundations of soft case-based reasoning*, volume 8. John Wiley & Sons, 2004.

[15] J. A. Recio-Garía and B. Díaz-Agudo. Ontology based cbr with jcolibri. In *International Conference on Innovative Techniques and Applications of Artificial Intelligence*, pages 149–162. Springer, 2006.

[16] M. T. Ribeiro, S. Singh, and C. Guestrin. Why should i trust you? explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1135–1144, 2016.

[17] D. Ross and W. D. Ross. *The right and the good*. Oxford University Press, 2002.

[18] T. Sato. A statistical learning method for logic programs with distribution semantics. In *ICLP*, 1995.

[19] S. Slade. Case-based reasoning: A research paradigm. *AI magazine*, 12(1):42–42, 1991.

[20] F. Sørmo, J. Cassens, and A. Aamodt. Explanation in case-based reasoning–perspectives and goals. *Artificial Intelligence Review*, 24(2):109–143, 2005.

[21] G. Van den Broeck, I. Thon, M. Van Otterlo, and L. De Raedt. Dtproblog: A decision-theoretic probabilistic prolog. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 24, 2010.