URN: urn:nbn:de:bvb:473-irb-466444
DOI: https://doi.org/10.20378/irb-46644

Künstliche Intelligenz manuscript No.
(will be inserted by the editor)

Please Delete That! Why Should I?

Explaining learned irrelevance classifications of digital objects

Michael Siebers · **Ute Schmid**

Received: date / Accepted: date

Abstract Dare2Del is an assistive system which facilitates intentional forgetting of irrelevant digital objects. For an assistive system to be helpful, the user has to trust the system's decisions. Explanations are a crucial component in establishing this trust. We will introduce different types of explanations which can vary along different dimensions such as level of detail and modality suitable for different application contexts. We will outline the cognitive companion system Dare2Del which is intended to support users managing digital objects in a working environment. Core of Dare2Del is an interpretable machine learning mechanism which induces decision rules to classify whether a digital objects is irrelevant. In this paper, we focus on irrelevance of files. We formalize the decision making process as logic inference. Finally, we present a method to generate verbal explanations for irrelevance decisions and point out how such explanations can be constructed on different levels of details. Furthermore, we show how verbal explanations can be related to the path context of the file. We conclude with a short discussion of the scope and restrictions of our approach.

Keywords Irrelevant digital objects · verbal explanations · Inductive Logic Programming

This work was funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – SCHM 1239/10-1 within the priority program SPP 1921 *Intentional Forgetting*.

M. Siebers
Cognitive Systems, University of Bamberg, Germany
E-mail: michael.siebers@uni-bamberg.de

U. Schmid
Cognitive Systems, University of Bamberg, Germany
E-mail: ute.schmid@uni-bamberg.de

1 Introduction

Digitalization, or the information age, has lead to an increase of stored data. While the world's total digital storage capacity (adjusted for optimal compression) was estimated at about 21.000 TB for 1997, that number grew to around 277 million TB in 2007 [14]. This explosion of information permeates our whole society, from data collected about individuals [44] over available research data and reports [15] to organisations and industry [7].

Research has shown that an *information overload* in the form of excessive amounts of digital objects (for example, files or emails) increases employees' stress levels, decreases job satisfaction, and hampers decision making [7,41]. In psychological research, it has been shown that unwanted effects of memory content can be decreased by intentional forgetting [3]. Intentional forgetting is a goal-directed mechanism to impair the recall of memory content unwanted or irrelevant for the current task. We propose to transfer the notion of intentional forgetting from mental representation to the world of digital objects. We assume that deletion of files and other digital objects can help individual employees as well as organisations to reduce information overload in work environments.

In an interdisciplinary collaboration with work and organizational psychology[1] we develop the prototypical assistive system Dare2Del which will support employees to manage digital objects more effectively. The system will in-

[1] Chair of Work and Organizational Psychology,
Friedrich-Alexander Universität Erlangen-Nürnberg,
`https://www.work-and-organizational-psychology.phil.fau.eu`.

fer if some digital object is irrelevant in the given (task) context. Then, this assessment can be exploited to assist the user in intentional forgetting. For example, Dare2Del might propose to delete or archive files or it might fade-out or hide task-irrelevant information [29,40]. In this paper, we will focus on the use case of permanent deletion of irrelevant files. Dare2Del is designed with the intention to reduce cognitive load for the user by pre-selecting candidate files for deletion. Dare2Del is designed as a companion system [9] that is human and system work together to realize a shared goal, namely efficient management of digital objects. System decisions are made transparent by allowing the user to demand an explanation for the system's proposal. The final decision whether a proposed file is really irrelevant and should thus be deleted lies with the user. User feedback is used to incrementally adapt the decision model of Dare2Del to individual criteria of irrelevance.

For a system to be supportive for a user it is necessary that system proposals are trusted [31]. This is true for knowledge based systems [4,46] as well as for machine learned classifiers [28,34]. One way to promote trust is explaining to the user why Dare2Del concludes that some file is irrelevant. Explanations have been researched extensively in the context of knowledge based systems [4,35,43]. However, the topic of explanation generation has only recently been recognized as relevant in the context of machine learning [21].

A core property of digital objects is that they relate to each other. For example, an email has several recipients, a phone number belongs to a contact, which in turn is in some address book, a file is older than another file, and a file is stored in a directory which belongs to a drive. Such kinds of relations can naturally be represented in first order logic. A logic framework also allows to make use of powerful inference mechanisms. For example, a transitivity rule could be exploited for such relations as *older* or *includes*.

In the next section, we will detail how irrelevance decisions are taken and how the necessary knowledge is modelled. In Section 3 we briefly describe two Inductive Logic Programming systems to provide a short intuition how the knowledge necessary may be generalized from decision examples. In Section 4 we localize our research within related work on explanations. Additionally, we describe how Dare2Del explains its decisions using natural language. Finally, we conclude with a discussion how to evaluate and improve our approach.

2 Modelling Relational Irrelevance Classifications

As mentioned above, Dare2Del pre-selects candidate files for deletion. Files are only proposed if the system decides that they are irrelevant. Before detailing the underlying decision process, we will give a summary of the used notation.

2.1 Terminology

Variables, predicate symbols, and function symbols are represented as string of letters, numbers, and underscores where variables must start with an upper case letter and predicate and function symbols with a lower case letter. The arity of a function or predicate symbol is the number of arguments it takes. A function which takes zero arguments is called a constant.

Every variable is a *term*. A function symbol of arity n followed by a bracketed n-tuple of terms is also a term. For constants, empty brackets may be omitted. A variable free term is called *ground*. A predicate symbol of arity n followed by a bracketed n-tuple of terms is called an *atom*, or a *positive literal*. The negation of an atom is called a *negative literal*. The negation symbol is not. A literal is called ground if all terms in its n-tuple are *ground*.

A *clause* is an implication where the antecedent is a conjunction of literals and the consequent is an atom. We write the implication reversed, as $H \leftarrow L_1 \wedge \ldots \wedge L_m$. If the conjunction of literals is empty ($m = 0$), we write $H \leftarrow \top$. The consequent of a clause C is called its *head*, $\text{head}(C)$. Its antecedent is called the *body* of the clause, $\text{body}(C)$. If $\text{body}(C) = \top$, we call C a *fact* and say that its body is *empty*. A clause is called ground when all its literals are ground. A set of clauses is called a *(clausal) theory*.

A substitution is a complete mapping from variables to terms. We denote a substitution by $\{x_1 \mapsto t_1, \ldots, x_k \mapsto t_k\}$ where $x_1, \ldots, x_k$ are variables and $t_1, \ldots, t_k$ are terms. A substitution is applied to a term by simultaneously replacing all x_i in the term by the corresponding t_i's. A substitution is applied to a literal by applying it to all terms in the literal. A substitution is applied to a clause by applying it to all literals in the clause. We denote the application of the substitution θ to a term, literal, or clause X by $X\theta$.

If some literal or conjunction of literals K is true given a clausal theory T, we say that T models K, or $T \models K$. Theory T models an atom A if there exists a clause $C \in T$ and a substitution θ such that $A = \text{head}(C)\theta$ and $T \models \text{body}(C)\theta$. Using negation by failure, a clausal theory T models every negative literal not A unless T models A. A theory T models

Table 1 Excerpt of data predicate symbols available to Dare2Del. Shown are example atoms with their descriptions.

Atom	Description
$\texttt{access\_time}(I,T)$	Item I was last accessed at time T.
$\texttt{change\_time}(I,T)$	Item I was last changed at time T.
$\texttt{in\_directory}(I,D)$	Item I resides in directory D.
$\texttt{media\_type}(F,M)$	File F contains data of the media type M.
$\texttt{name}(I,N)$	Item I has the name N.
$\texttt{newer}(F_1,F_2)$	File F_1 was created after file F_2.
$\texttt{similarity}(F_1,F_2,S)$	File F_1 and F_2 have a similarity of S.
$\texttt{size}(F,S)$	File F has S bytes.

a conjunction of literals $L_1 \wedge \ldots \wedge L_n$ if there is a substitution θ such that T models $L_i\theta$ for $1 \leq i \leq n$. By definition, the empty conjunction $\top$ is true given any theory.

2.2 Irrelevance Decisions

In order to infer the irrelevance of files, Dare2Del uses two clausal theories, one to represent the file system which may contain irrelevant files (*background knowledge*) and one to model irrelevance.

Background knowledge consists of information on *files* and *directories*, collectively referred to as *item*s. Every item can be characterized by a set of attributes—like its name (as usually displayed in a file manager), the time it was last accessed, or its size—and its relations to other items—for example, which directory a file is in or if files are similar. Items are represented by unique identifiers. Tables 1 and 2 contain some predicate symbols available to the system along with their interpretation.

Background knowledge is split in two parts: facts encoding information on individual objects (*data*, B_D), like the distance between files, and general clauses building on those facts (*domain theory*, B_T). For example, one file is larger than another if its size is greater than the other file's size. This separation is similar to knowledge representation in description logics which is split into assertional knowledge (ABox) and terminology (TBox) [1]. An excerpt of domain theory clauses can be seen in Figure 1.

Additional to the background theory, Dare2Del has a clausal theory for irrelevance $\mathbf{I}$. This theory mainly contains clauses with the head $\texttt{irrelevant}(F)$ for some variable F. Additionally, $\mathbf{I}$ may contain auxiliary clauses where the head's predicate symbol is neither `irrelevant` nor used in the background theory. Dare2Del considers a file with unique identifier f irrelevant if and only if $B_D \cup B_T \cup \mathbf{I} \models \texttt{irrelevant}(f)$. Some example irrelevance theory clauses are shown in Figure 2.

Table 2 Excerpt of domain theory relations. Shown are example atoms with their descriptions.

Atom	Description
$\texttt{accessed\_ago}(I,S)$	Item I was last accessed S seconds ago.
$\texttt{common\_infix}(F_1,F_2,I)$	I is a common infix of the names of files F_1 and F_2.
$\texttt{filename\_extension}(F,E)$	File F's name has extension E.
$\texttt{in\_same\_directory}(F_1,F_2)$	Files F_1 and F_2 are in the same directory.
$\texttt{larger}(F_1,F_2)$	File F_1 has a higher file size than File F_2.
$\texttt{not\_accessed\_since}(F,W)$	File F was not accessed within the last W seconds.
$\texttt{prefix\_length}(F_1,F_2,L)$	The names of files F_1 and F_2 start with the same L letters.
$\texttt{very\_similar}(F_1,F_2)$	Files F_1 and F_2 are very similar to each other.

$$\texttt{in\_same\_directory}(F_1,F_2) \leftarrow \texttt{in\_directory}(F_1,D) \wedge \texttt{in\_directory}(F_2,D)$$
$$\texttt{larger}(F_1,F_2) \leftarrow \texttt{size}(F_1,S_1) \wedge \texttt{size}(F_2,S_2) \wedge >(S_1,S_2)$$
$$\texttt{not\_accessed\_since}(F,W) \leftarrow \texttt{accessed\_ago}(I,S) \wedge >(S,W)$$
$$\texttt{prefix\_length}(F_1,F_2,L) \leftarrow \texttt{name}(F_1,N_1) \wedge \texttt{name}(F_2,N_2) \wedge \texttt{prefix}(N_1,L,Sub) \wedge \texttt{prefix}(N_2,L,Sub)$$

Fig. 1 Example clauses from domain theory. $\texttt{prefix}(S_0,L,S_1)$ holds if S_1 is a prefix of S_0 with length L. $>$ is the usual ordering on numbers.

$$\texttt{irrelevant}(F) \leftarrow \texttt{not\_accessed\_since}(F,31536000) \quad (1)$$
$$\texttt{irrelevant}(F) \leftarrow \texttt{in\_same\_directory}(F,F_2) \wedge \texttt{very\_similar}(F,F_2) \wedge \texttt{newer}(F_2,F) \wedge \texttt{prefix\_length}(F,F_2,5) \quad (2)$$
$$\texttt{irrelevant}(F) \leftarrow \texttt{in\_directory}(F,D) \wedge \texttt{name}(D,\text{"}tmp\text{"}) \quad (3)$$

Fig. 2 Example irrelevance theory clauses

3 Inductive Logic Programming for Learning Relational Irrelevance Classifications

Dare2Del may be equipped with a hand-crafted irrelevance theory. For example, a company might rule, that every copy of a file from the corporate network drive which has not been accessed within the last 30 days is irrelevant. Though such theories are useful, there is a need to learn irrelevance theories. On the one hand, rules may be too complicated to be manually expressed as clauses. On the other hand, the com-

panion system should adapt to the user's individual criteria of irrelevance.

In both cases, an irrelevance theory must be learned from given irrelevance decisions on individual files. As usual in machine learning, these decisions are called *examples* where irrelevant files are *positive examples* and non-irrelevant files are *negative examples*. In order to adapt to the user's criteria, these decisions are elicited by Dare2Del: given some initial (possibly hand-crafted) irrelevance theory Dare2Del decides that some files are irrelevant and proposes to delete these. Then, the user decides which of these to delete. We assume that this decision is solely based on the irrelevance of the file. Thus, deleted files form positive examples and not deleted files form negative examples for irrelevance. Formally, we represent positive examples and negative examples as sets of ground atoms.

Dare2Del uses Inductive Logic Programming methods to learn irrelevance theories. Inductive Logic Programming is a sub-field of symbolic machine learning which deals with learning clausal theories from examples. Additionally to examples, a background knowledge theory is provided to Inductive Logic Programming algorithms. The learned irrelevance theory together with the background knowledge theory must model all positive examples and no negative examples. Since introducing the whole field goes beyond the scope of this paper, we will shortly introduce two example Inductive Logic Programming systems with regard to learning irrelevance theories. Extensive overviews over the field are provided by De Raedt and Muggleton [6, 26].

3.1 Aleph

Aleph is a highly configurable software suite which is able to emulate several Inductive Logic Programming systems [42]. In the basic variant, Aleph learns a theory by learning one clause after the other. Each clause is learned by iteratively appending literals to the clause body.

Theory learning follows a user-defined language bias, limiting the set of possible theories. The language bias is given as so called *mode definitions* [25] and restricts the arguments a literal appended to the body may take. On the one hand, a type is assigned to every argument. On the other hand, argument terms must either be input, output, or constant. Output terms are variables which do not appear in any literal left of the one under consideration where every variable may only be used for a single type. Input terms are variables which appear in some body literal left of the one under consideration or in the head. Constant terms must be ground. For example, if `size` may take as first argument an input term of type *file* and as second argument an output term of type *number*, $\texttt{size}(F,S)$ is a valid literal to extend $\texttt{irrelevant}(F) \leftarrow \top$. In contrast, $\texttt{size}(G,S)$ and $\texttt{size}(F,1024)$ are not valid since G did not appear before and 1024 is not a variable.

To learn a single clause, Aleph first selects a positive example as seed. For that, it iteratively constructs the *most-specific clause* from literals adhering to the mode definitions. That is, assuming $\texttt{irrelevant}(f)$ is selected as seed, Aleph starts with the fact $\texttt{irrelevant}(f) \leftarrow \top$. Then, all atoms modelled by the background theory using only f as input term are collected. All terms substituted for output variables are extracted. In the next iteration, f and the extracted terms may be used as input terms. The process iterates a pre-specified number of times. Finally, f and all extracted terms within the literals are replaced by variables such that each variable corresponds to a given type in the mode definitions. Given the most-specific clause, a clause which models at least one positive example (the seed) and no negative example is searched. The body of this clause may only contain literals from the body of the most-specific clause and must adhere to the mode definitions. Finally, all positive examples modelled by the new clause are removed from the set of positive examples. If no more positive examples remain, the algorithm terminates. Otherwise, the next clause is constructed.

3.2 Metagol

Unlike Aleph, Metagol [5, 27] does not learn clauses independently one after another. Instead it learns a set of clauses simultaneously. As language bias, every single clause must follow a template, called *metarule*. Every metarule gives the number and arity of literals in the clause body and explicitly states their argument terms which may contain or be variables. Variables are either explicitly existentially quantified or implicitly universally quantified. Valid clauses are constructed from metarules by finding predicate symbols of appropriate arity and substituting existentially quantified variables by ground terms. For instance, Metagol could learn the clause $\texttt{irrelevant}(A) \leftarrow \texttt{size}(A,1024)$, when provided with the metarule $\exists B.P(A) \leftarrow Q(A,B)$. However, if this is the only available metarule, $\texttt{irrelevant}(A) \leftarrow \texttt{size}(A,B)$ and $\texttt{irrelevant}(A) \leftarrow \texttt{size}(A,1024) \wedge \texttt{name}(A,C)$ may not be learned since B must be substituted by a constant and the clause body may only contain one literal. But usually there is more than one metarule available to Metagol.

Before Metagol can learn a theory, the user has to fix the maximal number of clauses that shall be learned. Then, Metagol searches for a theory modelling all positive examples in a depth-first fashion. If some positive example is not modelled by the current theory and the maximal number of clauses is not reached, Metagol selects a metarule to follow, selects which predicate symbols to use and which constants to substitute for, and adds the resulting clause to the theory. When all positive examples are modelled, it tests whether any negative example is modelled. If this is not the case, Metagol terminates successfully. Otherwise, it backtracks, trying the next possible predicate symbols, constants, and metarules. Furthermore, Metagol invents additional auxiliary predicate symbols which may be used when constructing clauses from metarules. Thus, more complex rules can be build up from several simple clauses/metarules.

4 Explaining Relational Irrelevance Classifications

After learning an irrelevance theory, Dare2Del can apply this theory to determine which files are irrelevant and suggest the deletion of these files to the user. In the following, we will present how Dare2Del explains the suggestions to the user. First, we will define and and categorize explanations. Then, we detail which type of explanations are employed in Dare2Del and present function to generate such explanations. Finally, we show how explanations may be presented to the user and how interactive learning will be realized.

4.1 Dimensions of Explanations

In human-human interaction, if a person proposes an action to take or utters an opinion which we do not understand, we typically ask for an explanation. An explanation is defined as the act of making something clear by giving a detailed description, a reason, or a justification [20]. By explication, the line of reasoning of the other person becomes transparent and comprehensible to us. We can decide whether the explanation is convincing and consequently whether we will follow the given advice or not [30]. The relevance of explanations in human-computer interaction has been recognized over a long time in different domains. To our knowledge the first proposal for automated explanation generation has been made by Clancy [4] in the context of expert systems. He introduced a mechanism for collecting the trace of rule applications during inference of a diagnosis.

4.1.1 Functions of Explanations

Explanations have been studied quite extensively in the context of recommender systems. Tintarev [46] pointed out the different pragmatic functions of explanations: (a) providing *transparency* by detailing how the system reached some decision, (b) providing *scrutability* by allowing the user to review and correct system decisions, (c) increasing *trust*, and (d) increasing *effectiveness* by supporting good user decisions. The first function is directly addressed by Clancey's approach. The second function is in accordance with the notion of companion systems [2,9,40]. In contrast to AI research aiming at replacing humans by (super-)intelligent systems [13], research on companion systems follows the goal to provide assistive systems which support humans to deal with the ever increasing complexity of everyday life and of work environments. When such a companion system incorporates machine learning, there are specific demands, especially that learning has to be incremental and interactive [8,40].

That trustworthiness is crucial for acceptance of AI systems in many application domains has been recognized more and more during the last years, especially in the context of black-box classifiers induced by (deep) neural networks [13, 22]. One of the most well-known approaches is the LIME system [34] which provides explanations for classification decisions by providing local linear approximations. In the context of image classification, explanations are given by highlighting the areas in the input images which contributed most to the classifier decision. In the case of text classification, explanations are given as bag of relevant words of the input text.

That explanations are crucial in inducing trust has also been shown in the context of interpretable machine learning, that is, machine learning approaches where the learned classifiers are represented symbolically. Examples for such approaches are decision tree and decision set learners [10, 18] as well as inductive programming [12,39]. That providing classification rules in symbolic form results in more effectiveness has been shown in the context of inductive logic programming [28]. For a fictitious chemistry domain, users who had to induce the characteristics of a chemical substance from some positive and negative examples on their own classified significantly less test cases correctly as users who have been provided with the rules which have been learned from the initial examples. One of the pioneers of machine learning, Donald Michie had proposed that besides predictive accuracy operational effectiveness is an important evaluation criterion for machine learning approaches [24].

Approaches to explainable AI, such as LIME [34], provide explanations of the classification decision for a current instance, that is, they answer the question "Why is instance X classified as Y?", for example, "Why is this image classified as a dog?" or "Why is this file classified as irrelevant?" In contrast, approaches to interpretable machine learning address the comprehensibility of the learned classifier itself, that is, they answer the question "What is Y?", for example, "What is a dog?" or "What is an irrelevant file?" Interpretable machine learning thereby is strongly related to cognitive theories of categorization [17,50]. In the following, we will discuss why a restriction to one type of explanations—for instance, visual highlighting—often will not be adequate and why an explanation is not the same as interpretable machine learning.

4.1.2 Types of Explanations

Humans are adept to provide different kinds of explanations depending on the situational context or the assumed mental state of the recipient of the explanation. The relative merit of providing examples versus general rules as explanation has been researched extensively in educational contexts [19,33,37,49]. In research on analogical reasoning, it has been observed that for generalizing the relevant structure of a concept, it is helpful to align an example with a structurally similar example which does not belong to a category [23]. This same principle has been proposed in early AI by Winston [48] for learning structural descriptions from near misses. This idea has recently come back into focus within deep learning with the concept of generative adversarial networks [11]. In the context of recommender systems, explanations addressing different aspects of a decision have been discussed. For instance, a film recommender system investigated by Tintarev [45] provides explanations which can be generic (e.g., "this film is one of the top 100"), feature-based (e.g., "this film belongs to the genre drama"), or personalised feature-based (e.g., "this film features your favourite actor"). Depending on the user's cognitive abilities, time pressure, and emotional state, explanations might be more or less detailed [16].

Under the label of explainable AI currently mostly visual explanations are discussed [22,34] which is due to the fact that the most well-known and successful applications of black-box (deep) learning approaches are for image classification. In contrast, explanations in the context of rule-based inference and learning systems focus on explanations in form of rules—be it explication of a reasoning trace [4] or presentation of the learned classifier [28]. Supported by evidence on concept learning in the tradition of the dual code theory [36], we believe that combining visual and symbolic explanations can be helpful. Given a highly expressive symbolic language such as first order logic, explanations can grasp relevant information which cannot be directly transported in images. For instance, a relation such as "the stone is left of the stick" cannot be distinguished from a conjunction "there is a stone and a stick" using only visual highlighting. If the absence of an object is crucial for a classification, highlighting cannot provide this information [38]. There is a recent proposal to combine LIME with inductive logic programming to generate more expressive explanations [32].

4.1.3 Explanations in Dare2Del

In the context of Dare2Del, we apply inductive logic programming as a highly expressive approach to learn rules to classify whether digital objects are irrelevant. Our current focus is the generation of verbal explanations for irrelevance decisions concerning files. Similar to Clancey's approach [4], we present explanations in form of traces. However, in our context, the rules are learned from examples and not pre-defined. Furthermore, a trace is not presented in the given (Prolog) representation but rewritten in a natural language explanation. We address several aspects of explanation generation discussed above. First, explanations can be given in different detail: The most shallow explanation is to only take into account the relations given in the body of the rule which evaluated to be true for the file under consideration. In a next step, the relations in the body can be expanded by their respective rule definitions. Expansion terminates with the ground facts. By negating specific facts in such a way that the resulting instance cannot be classified by any of the classification rules, counterexamples could be constructed.

Such verbal explanations can be combined with specifics of the domain under discourse. In the context of application domains where the domain under discourse is based on images, for example aerial views of grave sites [32] or facial expressions of pain [38], we combine textual explanations with this visual information. In the context of irrelevance decisions for files, we relate the verbal explanation with specific views of the file system as domain of discourse. We believe, that grounding textual explanations in this way can heighten trust stronger than the verbal explanation alone.

$$\mathbf{T}_{\texttt{newer}}(s_1,s_2) = s_1 \cdot \texttt{" is newer than "} \cdot s_2$$
$$\mathbf{T}_{\text{not }\texttt{newer}}(s_1,s_2) = s_1 \cdot \texttt{" is as old as or older than "} \cdot s_2$$
$$\mathbf{T}^1_{\texttt{newer}}(t) = \texttt{"file "} \cdot n \quad \text{iff } B \cup \mathbf{I} \models \texttt{name}(t,n)$$
$$\mathbf{T}^2_{\texttt{newer}}(t) = \mathbf{T}^1_{\texttt{newer}}(t)$$

$$\mathbf{T}_{\texttt{accessed\_ago}}(s_1,s_2) = s_1 \cdot \texttt{" was accessed "} \cdot s_2 \cdot \texttt{" ago"}$$
$$\mathbf{T}_{\text{not }\texttt{accessed\_ago}}(s_1,s_2) = s_1 \cdot \texttt{" was not accessed "} \cdot s_2 \cdot \texttt{" ago"}$$
$$\mathbf{T}^1_{\texttt{accessed\_ago}}(t) = \mathbf{T}^1_{\texttt{newer}}(t)$$
$$\mathbf{T}^2_{\texttt{accessed\_ago}}(t) = \begin{cases} t/86400 \cdot \texttt{" days"} & \text{if } t \bmod 86400 = 0 \\ t/3600 \cdot \texttt{" hours"} & \text{if } t \bmod 3600 = 0 \\ t \cdot \texttt{" seconds"} & \text{otherwise} \end{cases}$$

$$\begin{aligned} \mathscr{T}_L\big(\texttt{newer}(file\_1, file\_2)\big) &= \mathbf{T}_{\texttt{newer}}\left(\mathbf{T}^1_{\texttt{newer}}(file\_1), \mathbf{T}^2_{\texttt{newer}}(file\_2)\right) \\ &= \mathbf{T}_{\texttt{newer}}(\texttt{"file foo"}, \texttt{"file bar"}) \\ &= \texttt{"file foo"} \cdot \texttt{" is newer than "} \cdot \texttt{"file bar"} \\ &= \texttt{"file foo is newer than file bar"} \end{aligned}$$

Fig. 3 Sample mapping functions for predicate symbols `newer` and `accessed_ago` and a sample application. It is assumed that the background knowledge contains clauses $\texttt{name}(file\_1, \texttt{"foo"}) \leftarrow \top$ and $\texttt{name}(file\_2, \texttt{"bar"}) \leftarrow \top$.

4.2 Generating Verbal Explanations

Dare2Del considers a file with identifier f as irrelevant if and only if $\texttt{irrelevant}(f)$ is modelled by its clausal theory. To explain this decision we transform the clause(s) and substitution(s) involved into a string in natural language. Given a finite set of symbols, called *alphabet*, a *string*, or *word*, is a finite sequence of symbols from the alphabet. Let the alphabet Σ be the set of Unicode characters. We will enclose strings within quotation marks, as in "this is a string". Let λ denote the empty string. The concatenation of two words w_1 and w_2 is denoted by $w_1 \cdot w_2$. The set of all strings over the alphabet Σ is denoted by $\Sigma^\star$.

For each predicate symbol p with arity n we define two functions from n words to a single word, $\mathbf{T}_\mathrm{p} : \Sigma^{\star n} \to \Sigma^\star$ and $\mathbf{T}_{\text{not p}} : \Sigma^{\star n} \to \Sigma^\star$. Additionally, we define a family of partial functions mapping ground terms to strings, $\left(\mathbf{T}^i_\mathrm{p}\right)_{1 \le i \le n} : G \to \Sigma^\star$ where G is the set of all ground terms. The functions $\mathbf{T}_\mathrm{p}$, $\mathbf{T}_{\text{not p}}$, and $\left(\mathbf{T}^i_\mathrm{p}\right)_{1 \le i \le n}$ can be seen as string templates for predicate symbol p and its arguments. An example template is shown in Figure 3. A string template is applied to a literal using the mapping function $\mathscr{T}_L : G \to \Sigma^\star$ defined as:

$$\mathscr{T}_L(L) = \begin{cases} \mathbf{T}_\mathrm{p}\left(\mathbf{T}^1_\mathrm{p}(t_1), \ldots, \mathbf{T}^n_\mathrm{p}(t_n)\right) & \text{if } L = \mathrm{p}(t_1, \ldots, t_n) \\ \mathbf{T}_{\text{not p}}\left(\mathbf{T}^1_\mathrm{p}(t_1), \ldots, \mathbf{T}^n_\mathrm{p}(t_n)\right) & \text{if } L = \text{not p}(t_1, \ldots, t_n) \end{cases}.$$

We extend the application from literals to a conjunction of literals, defining the function $\mathscr{T}_C$ as

$$\mathscr{T}_C(\top) = \lambda$$
$$\mathscr{T}_C(L) = \mathscr{T}_L(L) \quad \text{where } L \text{ is a literal}$$
$$\mathscr{T}_C(L_1 \wedge \ldots \wedge L_m) = \mathscr{T}_L(L_1) \cdot \texttt{" and "} \cdot \ldots \cdot \texttt{" and "} \cdot \mathscr{T}_L(L_m).$$

Finally, we can give an algorithm generating natural language explanations for irrelevance decisions. Given background knowledge B, an irrelevance theory $\mathbf{I}$, and a positive literal A, find a clause $C \in B \cup \mathbf{I}$ and a substitution θ such that $\text{head}(C)\theta = A$ and $B \cup \mathbf{I} \models \text{body}(C)\theta$. Without loss of generality we may assume that $C\theta$ is ground.[2] Then, $\mathscr{T}_L(A) \cdot \texttt{" because "} \cdot \mathscr{T}_C(\text{body}(C)\theta)$ is an explanation for A.

The combination of clause and substitution may be not unique. There may be more than one trace justifying the decision that a file is irrelevant. Given the irrelevance theory given in Figure 2 a file in folder "tmp" which was not accessed within the last year may be classified as irrelevant using clause (1) or clause (3). Consequently, the algorithm may produce more than one explanation.

4.3 An Explanation Interface

Figure 4 depicts a prototypic realization of Dare2Del. When suggesting the deletion of files, Dare2Del presents a list of multiple files which may be deleted. For each suggestion, the user is offered an explanation why the system considers this file as irrelevant. The explanation combines a verbalization of the reasoning trace with the file manager. The file under consideration and additional files or folders referenced in the explanation are highlighted in distinct colours in the text as well as in the file manager.[3]

The user can decide whether he or she wants to follow the system proposal or not. In some cases, the explanation will convince the user, in others not. In these cases, the user can override the system's classification and the system will modify the classifier by adapting the rules to the new information. While typically in such interactive learning scenarios [8] the interaction is restricted to a correction of the class, we believe that adaptation of the classification rules can profit by more information from the user. The user can mark parts in the explanations which are contradictory from

[2] If $C\theta$ is not ground we may substitute every variable in $C\theta$ with a unique constant.

[3] In order to realize coloured highlighting we designed the string templates to produce valid HTML strings. These are rendered by the displaying component.

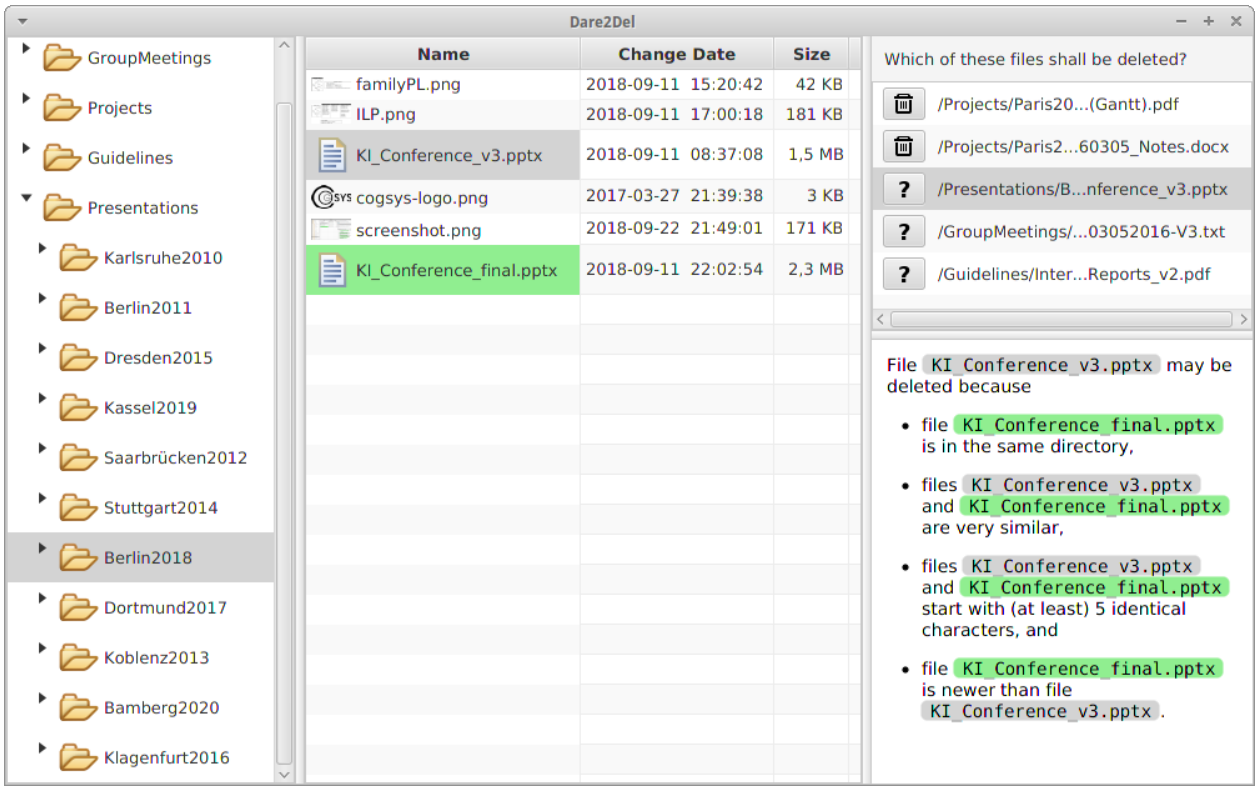

Fig. 4 Screenshot of deletion suggestions in Dare2Del. Deletion suggestions (top right) are shown together with an explanation why the selected file may be deleted (bottom right). Simultaneously, the relevant part of the file system is shown on the left. Colour highlightings are used to establish coherence between the explanation and the domain of discourse.

his or her perspective. That is, the interactive companion Dare2Del will be based on mutual explanations.

5 Conclusions

We outlined the cognitive companion system Dare2Del intended to support users managing digital objects in a working environment. We focused on irrelevance of files. First, we presented clausal theories as knowledge representation language. We distinguished background knowledge on individual objects, general domain background knowledge, and the irrelevance theory encapsulating clauses on the irrelevance of objects. Then we introduced Inductive Logic Programming to facilitate learning an irrelevance theory from user feedback. Finally, we discussed explanations in general and presented a method to generate verbal explanations for irrelevance decisions. We have shown how verbal explanations can be related to the path context of the file.

Though we have expounded the usefulness of explanations in general, we have not presented an evaluation of our approach. Regarding the usefulness of explanations in the Dare2Del file deletion scenario, we are currently working on a user study similar to the study by Wang & Benbasat [47]. There, we will investigate how verbal explanations and relating them to the domain of discourse influence the trust in a companion system. Evaluating the generated verbal explanations themselves is arduous, as there is no objective measure for the quality of explanations. We have to conduct a user study evaluating explanations generated by our approach regarding soft criteria, like comprehensibility or naturalness. In any case, we see one handicap of our approach: the one-to-one correspondence of terms to strings. Currently, the mapping only depends on the term itself. But, sentences using pronouns and relative clauses, like *"File a may be deleted because file b which is in the same folder is newer than it."*, do sound more natural. Realizing his would require keeping track of already verbalized terms and literals and referring back to them accordingly.

As an additional aspect, our trace-based approach could be extended to offer more general verbal explanations, similar to the proposal of Tintarev [45]. Explanations of irrel-

evance propositions could address general features, for example *this file is old*, be comparative, for example *yesterday file x, which is similar to the current file, was deemed irrelevant*, or user-specific, for example *you deleted several files like this one*.

Another line of research is strengthening the bond between verbal explanation and the file system context. Currently, we only highlight file and folder names. Depending on the predicates involved in the explanation, other information might be highlighted. For example, if `newer` is in the clause, the string `"is newer than"` and the change dates of the involved files could be highlighted.

References

1. Baader, F., Nutt, W.: Basic description logics. In: F. Baader, D. Calvanese, D. McGuinness, D. Nardi, P. Patel-Schneider (eds.) The Description Logic Handbook, pp. 43–95. Cambridge University Press (2003)
2. Biundo, S., Wendemuth, A.: Companion-technology for cognitive technical systems. KI – Künstliche Intelligenz **30**(1), 71–75 (2016)
3. Bjork, E.L., Bjork, R.A., Anderson, M.C.: Varieties of goal-directed forgetting. In: J.M. Golding, C.M. MacLeod (eds.) Intentional Forgetting: Interdisciplinary Approaches, vol. 103. Lawrence Erlbaum, Mahwah, NJ (1998)
4. Clancey, W.J.: The epistemology of a rule-based expert system – a framework for explanation. Artificial Intelligence **20**(3), 215–251 (1983)
5. Cropper, A., Muggleton, S.H.: Metagol system. URL https://github.com/metagol/metagol
6. De Raedt, L.: Logical and Relational Learning. Springer (2008). DOI 10.1007/978-3-540-68856-3
7. Eppler, M.J., Mengis, J.: The concept of information overload: A review of literature from organization science, accounting, marketing, MIS, and related disciplines. The Information Society **20**(5), 325–344 (2004). DOI 10.1080/01972240490507974
8. Fails, J.A., Olsen Jr., D.R.: Interactive machine learning. In: Proceedings of the 8th International Conference on Intelligent User Interfaces, pp. 39–45. ACM (2003)
9. Forbus, K.D., Hinrichs, T.R.: Companion cognitive systems – a step toward human-level AI. AI Magazine, special issue on Achieving Human-Level Intelligence through Integrated Systems and Research **27**(2), 83–95 (2006)
10. Fürnkranz, J., Kliegr, T., Paulheim, H.: On cognitive preferences and the interpretability of rule-based models (2018). Preprint, arXiv:1803.01316 [cs.LG]
11. Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative adversarial nets. In: Z. Ghahramani, M. Welling, C. Cortes, N.D. Lawrence, K.Q. Weinberger (eds.) Advances in Neural Information Processing Systems 27, pp. 2672–2680. Curran Associates, Inc. (2014). URL http://papers.nips.cc/paper/5423-generative-adversarial-nets.pdf
12. Gulwani, S., Hernandez-Orallo, J., Kitzelmann, E., Muggleton, S.H., Schmid, U., Zorn, B.: Inductive programming meets the real world. Communications of the ACM **58**(11), 90–99 (2015)
13. Hengstler, M., Enkel, E., Duelli, S.: Applied artificial intelligence and trust—the case of autonomous vehicles and medical assistance devices. Technological Forecasting and Social Change **105**, 105–120 (2016)
14. Hilbert, M., López, P.: The world's technological capacity to store, communicate, and compute information. Science **332**(6025), 60–65 (2011)
15. Huth, E.J.: The information explosion. Bulletin of the New York Academy of Medicine **65**(6), 647–672 (1989)
16. Jameson, A., Schäfer, R., Weis, T., Berthold, A., Weyrath, T.: Making systems sensitive to the user's changing resource limitations. Knowledge-Based Systems **12**(8), 413–425 (1999)
17. Kruschke, J.K.: Models of categorization. In: R. Sun (ed.) The Cambridge Handbook of Computational Psychology, pp. 267–301. Cambridge University Press (2008)
18. Lakkaraju, H., Bach, S.H., Leskovec, J.: Interpretable decision sets: A joint framework for description and prediction. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 1675–1684. ACM (2016)
19. Lombrozo, T.: Explanatory preferences shape learning and inference. Trends in Cognitive Sciences **20**(10), 748–759 (2016)
20. Lombrozo, T., Vasilyeva, N.: Causal explanation. In: M. Waldmann (ed.) Oxford Handbook of Causal Reasoning, pp. 415–432. Oxford University Press (2017)
21. Loza Mencía, E., Fürnkranz, J.: Interpretable machine learning. In: ECDA (ed.) Book of Abstracts, 5th European Conference on Data Analysis, pp. 56–60 (2018). URL http://groups.uni-paderborn.de/eim-i-fg-huellermeier/ecda2018/downloads/ECDA2018-BoA.pdf
22. Marcus, G.: Deep learning: A critical appraisal (2018). Preprint, arXiv:1801.00631v1 [cs.AI]
23. Markman, A.B., Gentner, D.: Commonalities and differences in similarity comparisons. Memory & Cognition **24**(2), 235–249 (1996)
24. Michie, D.: Machine learning in the next five years. In: Proceedings of the Third European Working Session on Learning, pp. 107–122. Pitman (1988)
25. Muggleton, S.: Inverse entailment and Progol. New Generation Computing **13**(3–4), 245–286 (1995)
26. Muggleton, S., De Raedt, L.: Inductive logic programming: Theory and methods. Journal of Logic Programming **19–20**, 629–679 (1994)
27. Muggleton, S.H., Lin, D., Tamaddoni-Nezhad, A.: Meta-interpretive learning of higher-order dyadic datalog: Predicate invention revisited. Machine Learning **100**, 49–73 (2015). DOI 10.1007/s10994-014-5471-y
28. Muggleton, S.H., Schmid, U., Zeller, C., Tamaddoni-Nezhad, A., Besold, T.: Ultra-strong machine learning: comprehensibility of programs learned with ILP. Machine Learning **107**(7), 1119–1140 (2018). DOI 10.1007/s10994-018-5707-3
29. Niessen, C., Göbel, K., Siebers, M., Schmid, U.: Time to forget: A review and conceptual framework of intentional forgetting in the digital world of work. Zeitschrift für Arbeits- und Organisationspsychologie [German Journal of Work and Organizational Psychology] (to appear)
30. Potter, J., Wetherell, M.: Discourse and Social Psychology: Beyond Attitudes and Behaviour. Sage (1987)
31. Pu, P., Chen, L.: Trust-inspiring explanation interfaces for recommender systems. Knowledge-Based Systems **20**(6), 542–556 (2007)

32. Rabold, J., Siebers, M., Schmid, U.: Explaining black-box classifiers with ILP – empowering LIME with Aleph to approximate non-linear decisions with relational rules. In: F. Riguzzi, E. Bellodi, R. Zese (eds.) Proceedings of the 28th International Conference on Inductive Logic Programming, pp. 105–117 (2018)
33. Reed, S.K., Bolstad, C.A.: Use of examples and procedures in problem solving. Journal of Experimental Psychology: Learning, Memory, and Cognition **17**(4), 753–766 (1991)
34. Ribeiro, M.T., Singh, S., Guestrin, C.: "Why should I trust you?": Explaining the predictions of any classifier. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 1135–1144. ACM (2016). URL http://arxiv.org/abs/1602.04938
35. Roth-Berghofer, T., Richter, M.M.: Schwerpunkt: Erklärungen [Special issue: Explanations]. KI – Künstliche Intelligenz **22**(2) (2008)
36. Sadoski, M., Paivio, A.: Imagery and Text: A Dual Coding Theory of Reading and Writing. Routledge (2013)
37. Schmid, U.: Programmieren lernen: Unterstützung des Erwerbs rekursiver Programmiertechniken durch Beispielfunktionen und Erklärungstexte [Learning programming: Acquisition of recursive programming skills from examples and explanations]. Kognitionswissenschaft **4**(1), 47–54 (1994)
38. Schmid, U.: Inductive programming as approach to comprehensible machine learning. In: C. Beierle, G. Kern-Isberner, M. Ragni, F. Stolzenburg, M. Thimm (eds.) Proceedings of the 7th Workshop on Dynamics of Knowledge and Belief (DKB-2018) and the 6th Workshop KI & Kognition (KIK-2018), co-located with 41st German Conference on Artificial Intelligence, vol. 2194. CEUR Workshop Proceedings (2018)
39. Schmid, U., Kitzelmann, E.: Inductive rule learning on the knowledge level. Cognitive Systems Research **12**(3), 237–248 (2011)
40. Siebers, M., Göbel, K., Niessen, C., Schmid, U.: Requirements for a companion system to support identifying irrelevancy. In: 2017 International Conference on Companion Technology, pp. 1–2 (2017). DOI 10.1109/COMPANION.2017.8287076
41. Soucek, R., Moser, K.: Coping with information overload in email communication: Evaluation of a training intervention. Computers in Human Behavior **26**(6), 1458 – 1466 (2010). DOI 10.1016/j.chb.2010.04.024
42. Srinivasan, A.: The aleph manual (2004). URL http://www.cs.ox.ac.uk/activities/machinelearning/Aleph/
43. Suthers, D.D.: An analysis of explanation and its implications for the design of explanation planners. Ph.D. thesis, University of Massachusetts (1993)
44. Sweeney, L.: Information explosion. In: L. Zayatz, P. Doyle, J. Theeuwes, J. Lane (eds.) Confidentiality, Disclosure, and Data Access: Theory and Practical Applications for Statistical Agencies, pp. 43–74. Urban Institute (2001)
45. Tintarev, N., Masthoff, J.: Evaluating the effectiveness of explanations for recommender systems. User Modeling and User-Adapted Interaction **22**(4), 399–439 (2012)
46. Tintarev, N., Masthoff, J.: Explaining recommendations: Design and evaluation. In: Recommender Systems Handbook, pp. 353–382. Springer (2015)
47. Wang, W., Benbasat, I.: Recommendation agents for electronic commerce: Effects of explanation facilities on trusting beliefs. Journal of Management Information Systems **23**(4), 217–246 (2007). DOI 10.2753/MIS0742-1222230410
48. Winston, P.H.: Learning structural descriptions from examples. In: P.H. Winston (ed.) The Psychology of Computer Vision, pp. 157–210. McGraw-Hill, . New York (1975)
49. Zeller, C., Schmid, U.: Automatic generation of analogous problems to help resolving misconceptions in an intelligent tutor system for written subtraction. In: A. Coman, S. Kapetanakis (eds.) Workshops Proceedings for the 24th International Conference on Case-Based Reasoning, vol. 1815, pp. 108–117. CEUR Workshop Proceedings (2016). URL http://ceur-ws.org/Vol-1815/paper11.pdf
50. Zeller, C., Schmid, U.: A human like incremental decision tree algorithm: Combining rule learning, pattern induction, and storing examples. In: M. Leyer (ed.) LWDA Conference Proceedings, vol. 1917, pp. 64–73. CEUR Workshop Proceedings (2017). URL http://ceur-ws.org/Vol-1917/paper12.pdf

Michael Siebers is research assistant in the Cognitive Systems Group at University of Bamberg. He holds a diploma in psychology and a B. Sc. in applied computer science, both from University of Bamberg. His main research interest is Inductive Logic Programming and human interpretable machine learning, especially for structured data.

Ute Schmid holds a diploma in psychology and a diploma in computer science, both from Technical University Berlin (TUB). She received her doctoral degree in computer science from TUB in 1994 and her habilitation in computer science in 2002. She worked as assistant and lecturer at TUB and University of Osnabrück and has been visiting researcher at Carnegie-Mellon University. Since 2004 she is professor for Applied Computer Science/Cognitive Systems at University of Bamberg. Her research focus is on symbolic/knowledge-level approaches of machine learning for structural data, especially inductive programming.