Siebers, Michael; Schmid, Ute; Göbel, Kyra; Niessen, Cornelia:

A Psychonic Approach to the Design of a Cognitive Companion Supporting Intentional Forgetting

In: Kognitive Systeme / 2017 - 1

DOI: http://dx.doi.org/10.17185/duepublico/44537

URN: urn:nbn:de:hbz:464-20170927-085543-7

Link: http://duepublico.uni-duisburg-essen.de/servlets/DocumentServlet?id=44537

A Psychonic Approach to the Design of a Cognitive Companion Supporting Intentional Forgetting ⋆

Michael Siebers, * **Ute Schmid,** * **Kyra Göbel,** ** **Cornelia Niessen** **

* *Cognitive Systems, Faculty of Information Systems and Applied Computer Science, University of Bamberg, Germany (e-mail: {michael.siebers,ute.schmid}@uni-bamberg.de).*
** *Chair for Work and Organizational Psychology, Faculty of Humanities, Social Sciences, and Theology, University of Erlangen, Germany (e-mail: {kyra.goebel,cornelia.niessen}@fau.de).*

Abstract: We present the design basics of the cognitive companion Dare2Del which supports humans to temporarily ignore or permanently forget digital objects. The system infers the irrelevancy of digital objects based on a symbolic knowledge representation and white box machine learning. The logic-like knowledge base of the system is split in facts that apply to individual digital objects (data) and general rules which hold for all objects in the domain (domain theory). The irrelevancy of digital objects is inferred using explicit rules, which are learned by an inductive logic programming approach.
In this paper, we focus on the knowledge base and the inference for the permanent deletion of files. First, we present relations available for data and domain theory. Then, we evaluate whether the chosen relations are suitable for inferring the irrelevancy of files. We evaluate a handcrafted irrelevancy rule on an artificial file system with 500 files. For that, we developed a generator for knowledge bases of artificial file systems. We were able to classify 97.2% of the files correctly as irrelevant or relevant.

Keywords: Companion System, Symbolic Knowledge Representation, White Box Machine Learning, Irrelevant Digital Objects

1. INTRODUCTION

Digitalization in industry and administration results in long-term storage of growing amounts of data. However, large amounts of potentially irrelevant digital objects can hinder efficient work and decision making of individual employees and negatively affect organizational performance: Employees can get distracted by irrelevant digital information; irrelevant or even obsolete data can result in inefficient procedures. This in turn increases the time needed to finish assignments, thus, increasing costs. In contrast to computer systems, the human cognitive system offers effective mechanisms for ignoring irrelevant information and for long-term decay of information (Bjork et al., 1998). These might be used as inspiration for the design of algorithmic forgetting.

In an interdisciplinary cooperation between psychology and computer science, we work on a prototypical realization of an assistive system (Dare2Del) which supports employees to manage digital content more effectively. It supports two different processes, (a) temporarily ignoring and (b) permanently deleting digital objects. In the first case, the presentation of information is faded out in a given context by the system. In the second case, the system proposes to delete (or archive) digital objects. Hence, the companion Dare2Del can be seen as a special kind of recommender system (Ricci et al., 2011). The design of Dare2Del is inspired by psychological theories and empirical findings concerning human mechanisms of forgetting. Thus, in analogy to bionics, our design strategy can be labelled as 'psychonics' (Schmid, 2008).

Context-sensitive fading out and permanent deletion of digital objects promotes employees to forget irrelevant information. Like human self-regulation which is known to be an important skill in adaptive task performance (Niessen and Jimmieson, 2016), this helps employees to focus their attention, avoid interference, and improve their decision making performance. Additionally, the externalization of the regulation process should decrease cognitive load. We argue that an assistant system is especially important for individuals who have difficulties to forget unwanted and irrelevant information, and in situations in which forgetting is hindered by factors such as daily hassles.

In empirical studies, we will address which situational demands at work help or hinder intentional forgetting of

⋆ This work is funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – SCHM 1239/10-1; NI 1066/3-1.

(internal) memory content or (external) digital objects. First, we will conduct an experience sampling study with administrative staff collecting data of critical events in which individuals at work deliberately forget and delete digital information (Niessen et al., 2017). In another experiment, we will investigate whether fading irrelevant information actually supports forgetting of irrelevant digital objects, with positive consequences for task performance and cognitive load.

Dare2Del is to be realized as a cognitive companion (Forbus and Hinrichs, 2006), that is, the interaction relies rather on mutual than asymmetric support. In order to have a system which is transparent to the user, we rely on explicit knowledge representation (Beierle and Kern-Isberner, 2014) in combination with explainable decision rules generated by white box approaches to machine learning (Schmid and Kitzelmann, 2011). Making the system generic, the domain specific knowledge will be separated in its own module facilitating easy adaptation to different application contexts. In our research, we focus on management of (a) documents in administration and (b) quality features in industrial production.

In the following, we first give an overview of the requirements for the companion system Dare2Del. Afterwards, we present a first case study focussing on deletion of different types of files. We present the logic model of files which represents the knowledge which a human might rely on when deciding whether a file is irrelevant and therefore can be deleted. Then, we give a proof-of-concept demonstrating that the irrelevancy of a file can be inferred subject to certain of its characteristics. We conclude with a short summary and further work to be done.

2. THE COGNITIVE COMPANION DARE2DEL

Dare2Del is conceptualized as a knowledge-based system using logical representations and incorporating different inference and learning mechanisms. It can be viewed as an expert companion to an individual in some working environment. The structure of Dare2Del and its interactions with a user are shown in Fig. 1. On the one hand, it shapes the digital environment such that the individual can focus its attention on the relevant task at hand. This will be realized by fading out irrelevant information, which makes system decisions transparent to the user—in contrast to hiding information. Alternatively, important information might be highlighted. On the other hand, it makes explicit proposals to delete (or archive) digital objects. The first behaviour is adaptive and implicit, that is, if Dare2Del is activated by a user, it will automatically fade out digital objects inferred to be irrelevant given a specific task context. The second behaviour, in contrast, is explicit, proposing to the user that a certain digital object might be permanently deleted (or archived), giving reasons for this proposal.

The design of Dare2Del is modular: the automated inference process for fading out and deleting are based on the same principles and applications to different domains—such as file management in administration and data management in industry 4.0—will be realized by exchanging the domain dependent knowledge while the general mechanisms are kept. Core of the system is to infer if a digital object I is irrelevant in some context C, represented by the predicate *irrelevant(I,C)*. If a digital object is inferred to be irrelevant in a specific context it can be temporarily faded out in that context. If information is inferred to be irrelevant in all contexts, the system can propose permanent deletion (or archiving).

Inference of irrelevancy in Dare2Del is based on symbolic knowledge representation. Digital objects under consideration are represented in a logic-like syntax. Then, the irrelevancy of an object is inferred using rules. These rules will be learned with an inductive logic programming (ILP) approach (Muggleton and De Raedt, 1994; Schmid et al., 2017). As the resulting rules are also represented on the knowledge level (white box machine learning), Dare2Del is able to inspect its own reasoning and can exploit this to generate explanations when interacting with humans. That is, Dare2Del will be realised as a knowledge-based system with an explanation component (Beierle and Kern-Isberner, 2014).

As a first realization of Dare2Del we focus on the administration domain and on permanent deletion of files. For example, if a pdf-file with the same name and same size exists in different directories, all except one of them could be deleted. However, which of these should be inferred to be irrelevant depends on idiosyncrasies. Similarly, if there are several versions of a file, many users delete the older versions. If a specific user agrees to this principle, a general predefined rule may be used. On the other hand, a user might prefer to save older versions if they contain text which is no longer included in newer versions. This might require a quite sophisticated rule, which must be learned. This example illustrates the need for a combination of predefined and learned rules. Although there are some proposals how to integrate learning from observation of application use (Linton and Schaefer, 2000), there exists no general solution to this problem.

3. USE CASE: IRRELEVANT FILE OBJECTS

As a first use case for Dare2Del, we modelled the irrelevance of files within a file system. To this end, humans typically rely on specific knowledge about the considered files, general knowledge about file systems, as well as experience. Hence, our knowledge base is split in facts that apply to individual objects (*data*) and general relations which hold for all objects in the domain (*domain theory*). That is, information gathered about individual items can be enriched by more general (background) knowledge which can also be exploited during inference and learning. In the following, we detail the available relations of both types along with their semantics. Since the intended system will use ILP at its core, we use a Prolog-like representation. An overview is provided in tables 1 and 2.

3.1 Data Relations

To represent our file system, we distinguish between *files* and *directories*.[1] Files and directories together are referenced as *items*. Each item is referenced by an unique identifier `I`. Files are denoted by `file(I)`, directories by

[1] We restrict ourselves to user-producible directories which may contain files. The more general concept of folders is omitted.

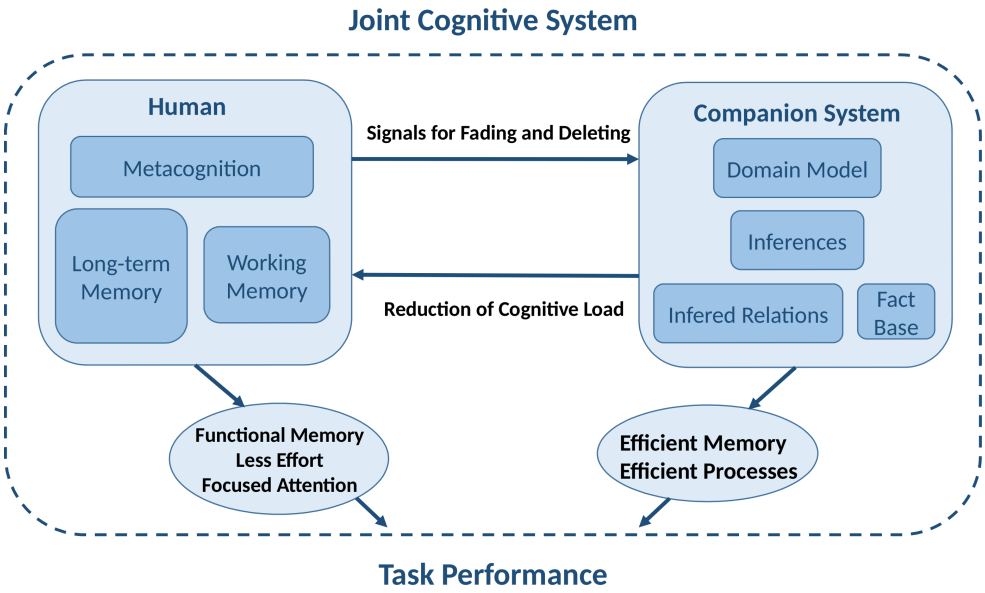

Fig. 1. Main components of the cognitive companion Dare2Del and its interaction with a human

Table 1. Summary of data relations

Name	Description
`access_time(I,T)`	Item `I` was last accessed at time `T`.
`byte_dist(A,B,D)`	Files `A` and `B` have a Levenshtein distance on bytes of `D`.
`change_time(I,T)`	Item `I` was last changed at time `T`.
`creation_time(I,T)`	Item `I` was created at time `T`.
`directory(I)`	Item `I` is a directory.
`file(I)`	Item `I` is a file.
`in(I,D)`	Item `I` resides in directory `D`.
`media_type(F,M)`	File `F` contains data of the media type `M`.
`modification_time(I,T)`	Item `I` was last modified at time `T`.
`name(I,N)`	Item `I` has the name `N`.
`size(F,S)`	File `F` has `S` bytes.

Table 2. Summary of domain theory relations

Name	Description
`accessed_ago(I,S)`	Item `I` was last accessed `S` seconds ago.
`byte_dist_norm(A,B,D)`	Files `A` and `B` have a relative Levenshtein distance on bytes of `D`.
`changed_ago(I,S)`	Item `I` was last changed `S` seconds ago.
`common_infix(A,B,I)`	`I` is a longest common infix of strings `A` and `B`.
`common_prefix(A,B,P)`	`P` is the longest common prefix of strings `A` and `B`.
`common_suffix(A,B,S)`	`S` is the longest common suffix of strings `A` and `B`.
`created_ago(I,S)`	Item `I` was created `S` seconds ago.
`filename_base(F,N)`	File `F`'s name has base name `N`.
`filename_extension(F,N)`	File `F`'s name has extension `N`.
`modified_ago(I,S)`	Item `I` was last modified `S` seconds ago.
`within(I,D)`	Item `I` is in directory `D` or in one of `D`'s sub-directories.

`directory(I)`. Every item has a name, which is usually shown in a file manager. The predicate `name(I,Name)`, states that the item `I` has the name `Name`. Every item has exactly one name.

Every item `I` has four associated times `T` (seconds since epoch): (a) the time it was created (`creation_time(I,T)`), (b) the time its content was changed (`change_time(I,T)`), (c) the time its meta-data—like its name—was altered (`modification_time(I,T)`), and (d) the time it was last accessed (`access_time(I,T)`). Every item has exactly one time information of each kind. [2]

The usual tree-like structure of file systems is modelled using the predicate `in`, where `in(I,D)` denotes that item `I` resides within directory `D`. Every file must reside within one directory. Every directory may reside within another directory. Thus, multiple file system roots are permitted.

For each file its size `S` in bytes (`size(F,S)`) and the type of its content is represented. The content is conveyed by the media type of the file, `media_type(F,media(T/S))`, where `T` is the top media type (such as `text` or `application`) and `S` is the subtype including tree (such as `plain` or `vnd.ms-excel`). [3] A complete representation of an example file is given in Fig. 2.

As mentioned above, it is necessary to define similarity measures (or distances) between files. These distances and similarities must be content dependent, since there is hardly any use in comparing a Microsoft Excel Sheet with a Bitmap Graphic. As this work is not about similarities and distances between files, we only use a simple distance measure for illustration, the Levenshtein distance (Yujian and Bo, 2007) using bytes as alphabet. Every pair of files `F1` and `F2` has an associated distance (`byte_dist(F1,F2,D)`, where `D` is a positive integer) if and only if both files have the same media type.

[2] We are aware that not every real-world file system does provide all four kinds of time information. However, for the purpose of this paper we assume that the information is fully available.
[3] Media type parameters are currently not represented.

```
file(item_11).
name(item_11,'meetingProtocol.txt').
media_type(item_11,media(text/plain)).
size(item_11,2287).
creation_time(item_11,1482580800.0).
change_time(item_11,1482580800.0).
modification_time(item_11,1482580800.0).
access_time(item_11,1483300800.0).
in(item_11,dir_1).
```

Fig. 2. Complete data representation of an example file: The file 'meetingProtocol.txt' is referenced by the identifier `item_11`. The file is a plain text file which was created at noon on Christmas Eve 2016 (represented as seconds since epoch), never modified, and last accessed at 8 p.m. on New Year's Day 2017. The file has 2,287 bytes and is in the directory referenced by `dir_1`.

3.2 Domain Theory Relations

Having information on specific files and directories as given above, we can automatically derive further information, for example whether a file is older than another, if they share some part of their names, or if a file resides in a sub-directory of some directory. This derived information is inferred using Prolog rules for additional relations. In the following, we present these relations, including their semantics.

Usually, file names can be split into a base name and an extension, where each file has a base name. The extension is optional. The predicate `filename_base(F,B)` states that the file with the identifier `F` has the base name `B`. The extension `E` for the same file is represented by `filename_extension(F,E)`. The period between base name and extension is not represented (outside of the predicate `name`). In order to handle similarities between item names, the domain theory is augmented by three predicates handling sub-strings. Given strings `A` and `B`, `common_prefix(A,B,P)` gives the one and only longest common prefix `P` between the two files, `common_suffix(A,B,S)` states that `S` is the one and only longest common suffix, and `common_infix(A,B,I)` states that `I` is one of the longest [4] common infixes.

We represent the time span `S` (as fraction of seconds) since some item's creation (`created_ago(I,S)`), since its last content change (`changed_ago(I,S)`), since its last modification (`modified_ago(I,S)`), and since it was last accessed (`accessed_ago(I,S)`). Furthermore, we build the transitive closure of the `in`-relation, `within(I,D)`, where `I` is the identifier of an item and `D` is the identifier of a directory.

To remedy the dependency of the Levenshtein distance on the sizes of the files, we introduce a normalization of this metric. Given `size(A,SA)` and `size(B,SB)`, the Levenshtein distance between `A` and `B` is between $|\mathtt{SA} - \mathtt{SB}|$ and $\max(\mathtt{SA}, \mathtt{SB})$. Thus, we define the normalised Levenshtein distance [5] (`byte_dist_norm(A,B,N)`) as linear transformation from this range to the interval from 0 to 1. [6]

Finally, `irrelevant(I)` represents that item `I` is irrelevant in all contexts and, thus, may be deleted. This relation is neither part of the data nor of the domain theory. Instead, this is the target predicate for which rules should be learned for the final companion system. Additionally to predicates introduces above, Prolog build-in predicates, like string length (`atom_length`) or negation (`\+`), may be used within these rules. For instance rule 1 from Fig. 3 states that a directory `D` is irrelevant if there is no item that resides inside it, that is if the directory is empty. Rule 2 on the other hand states that a file may be deleted if it has not been accessed within the last thirteen months. Finally, rule 3 states that some file `F` is irrelevant, when it has not been accessed since another file with the same media type was created in the same directory if both files' names share a prefix of length not less than five.

4. PROOF OF CONCEPT

Given our representation, we should be able to define rules which can distinguish between relevant and irrelevant items. To investigate that claim, we generated an artificial file system with relevant and irrelevant files and evaluated a hand-crafted `irrelevant(F)`-rule on it. In the following, we first describe the generation of random files and then evaluation results.

The file system is generated iteratively. In each iteration a batch of a random number of files is generated following a specific scenario. Every scenario is an instantiation of a specific pattern, where the most basic pattern is pure *random* generation. As contrasting pattern we selected the derivation of files from a parent file (*derived*).

In the *random* pattern, first, a parent directory for the file is chosen. This may either be an existing directory or a newly created one. Then, media type, base name, and an extension matching the media type is chosen. The size of the file is drawn randomly from a Gaussian distribution respecting a minimal and a maximal size. Finally, creation, change, modification, and access time are chosen in ascending order. A pseudo-code algorithm for random creation is given in algorithm 1. We created seven scenarios from this pattern using the media types *text/plain*, *application/msword*, *application/vnd.ms-excel*, *image/tiff*, *image/png*, and *application/pdf* with sensible extensions and file size constraints. For example, *text/plain*-files may have the extensions 'txt', 'TXT', 'ME', or none at all. The average file size is 2 KB with a variance of 100 bytes, a minimum of 1 byte and a maximum of 5 KB.

In the *derived* pattern, first, a parent file is drawn randomly from all known files. Media type and extension are assumed for the new files. The base names for the files are constructed by appending '_vNN' to the base name of the parent file, where NN are consecutive numbers. Each file is derived from the previous (starting from the parent file) by choosing the number of bytes to add to the file

[4] The longest common infix of two strings is not necessarily unique. The strings *car* and *rat*, for example, have two infixes of equal length in common, *a* and *r*. However, they have no longer common infix.

[5] Note that this normalization is no metric, since the triangle inequality does not hold. However, the normalization is symmetrical.

[6] In the degenerated case—the minimal distance equals the maximal distance—the normalization is defined as 0.

```
(1) irrelevant(D) :- directory(D), \+(in(AnyItem,D)).
(2) irrelevant(F) :- file(F), accessed_ago(F,Ago), Ago > 33696000.
(3) irrelevant(F) :- media_type(F,M), media_type(Other,M), in(F,Dir), in(Other,Dir),
    access_time(F,AccessF), creation_time(Other,CreationO), AccessF < CreationO,
    name(F,NameF), name(Other,NameO), common_prefix(NameF,NameO,Pre), atom_length(Pre,N), N >= 5.
```

Fig. 3. Example rules producible with the presented relations. The meaning of the rules is explained in the text.

Input: N: number of files to create
S, E: a set of media types, a set of suitable extensions
min, *max*: minimal and maximal file size
avg, *var*: average and variance of file size

with a probability of *25%* **do**
  draw *directory* uniformly random from all existing directories;
otherwise
  get new unique *id*;
  draw random alphanumeric *name* of length 8;
  create new *directory*;
end

for N *times* **do**
  get new unique *id*;
  draw *media type* uniformly random from S;
  draw *file name extension* uniformly random from E;
  draw random alphanumeric *base name* of length 8;
  draw *size* from $\mathcal{N}(avg,var)$ such that $min \leq size \leq max$;
  draw *creation time*, *change time*, *modification time*, and *access time* uniformly, such that
    start of epoch $\leq$ *creation time* $\leq$ *change time* $\leq$ *modification time* $\leq$ *access time* $\leq$ *now*;
  create *new file* and place it in *directory*;
end

Algorithm 1: Random creation of files in a random directory

Input: N: number of files to create
PChange, *PAdd*, *PDel*: average percentage of file to change, to be added, and to be deleted

draw *parent* file uniformly random from all existing files;
for i *from 2 to* N*+1* **do**
  base name $\leftarrow$ *parent*'s base name + "_v" + i;
  copy *extension* and *media type* from *parent*;
  draw *add* from *parent*'s size $* \mathcal{N}(PChange{+}PAdd,(PChange{+}PAdd)/5)$ where $0 \leq add$;
  draw *del* from *parent*'s size $* \mathcal{N}(PChange{+}PDel,(PChange{+}PDel)/5)$ where $0 \leq del \leq$ *parent*'s size;
  size $\leftarrow$ *parent*'s size + *add* $-$ *del*;
  draw *creation time*, *change time*, *modification time*, and *access time* uniformly, such that
    parent's *access time* $\leq$ *creation time* $\leq$ *change time* $\leq$ *modification time* $\leq$ *access time* $\leq$ *now*;
  create *new file* and place it in the directory containing *parent*;
  draw *distance* between *new file* and *parent* uniformly such that $\max($*size*, *parent*'s size$) \leq$ *distance* $\leq$ *add* + *del*;
  parent $\leftarrow$ *new file*;
end

Algorithm 2: Derived creation of files

and the number of bytes to delete from the file. Both are drawn from a Gaussian distribution based on the size of the previous file. The size of the new file is computed accordingly. The Levenshtein distance between the files is uniformly drawn within the theoretical bounds. This pattern is shown as pseudo-code in algorithm 2. We used three scenarios from this pattern: revising, shortening, and extending files.

Finally, after creating all files, the unknown Levenshtein distances are drawn uniformly within the theoretical bounds such that the triangle inequality holds.

For the evaluation we generated a data set with 500 files, 69% of which were generated *random*, 31% were *derived*. The files reside in 58 directories, with an average of 3.17 directories above a file. There are 60 *text/plain*-, 69 *application/msword*-, 85 *application/vnd.ms-excel*-, 60 *image/tiff*-, 136 *image/png*-, and 90 *application/pdf*-files.

Table 3. Confusion matrix of files used as *parent* in derived generation and files identified as *irrelevant* using rule 3 from Fig. 3.

		Irrelevant		
		yes	no	
Parent	yes	141	6	95.9%
	no	8	345	97.7%
		94.6%	98.3%	97.2%

On this data set we evaluated rule 3 from Fig. 3. We assume that all files which were *used as parent* in *derived* generation are superseded by the derived file and are thus irrelevant. Hence, the `irrelevant(F)`-relation should hold for all such files and for no other files. As table 3 shows, this claim is correct for 97.2% of the files.

5. CONCLUSIONS AND FURTHER WORK

We presented a first realization of Dare2Del based on a domain model for files and directories and a handcrafted rule to decide whether a file is irrelevant—and therefore can be deleted. We realized a generator for file systems which is parametrized with respect to the number of files to be generated. Media types and additional features such as size are generated based on probability distributions. In the current paper we used Gaussian distributions. For features such as size, a long tail distribution where (very) large sizes are less probable might be more realistic. We gave a first proof of concept to demonstrate that our explicit logic based approach to the identification of irrelevant digital objects is feasible.

The purely knowledge-based approach of Dare2Del can now be extended to incorporate probabilistic reasoning and learning. We plan to induce additional rules for irrelevance from a set of random file systems with the ILP system Metagol (Cropper and Muggleton, 2016). Since learning in ILP is supervised,that is, positive and negative examples are available, we plan to generate hypothetical user decisions where Dare2Del proposed to delete a file and the user either decided in favour or against this proposal. Probabilistic ILP (De Raedt and Kimmig, 2015) will be explored when addressing inference of task contexts.

Further, the (meta) information available to the system might be extended. We will incorporate additional information of files and directories available from the file system, like whether a directory is local or on a network storage or the users permissions on a file. We will integrate more sophisticated distance or similarity measures for specific media types. Additionally, immediate file dependencies, such as an HTML-document embedding an image, might be modelled.

In another line of research, we are conducting psychological studies and experiments to give insight into individual abilities and preferences to forget, and into the situational facilitators and constraints of forgetting. Managing human memory and external memory together with the cognitive system Dare2Del should help workers to concentrate and perform better on the task at hand—even in a context of large amounts of (irrelevant) digital data and information. Thus, Dare2Del should not only facilitate individual performance but also lead to further positive organizational consequences.

REFERENCES

Beierle, C. and Kern-Isberner, G. (2014). *Methoden wissensbasierter Systeme - Grundlagen, Algorithmen, Anwendungen*. Computational Intelligence. Springer Vieweg, 5th edition.

Bjork, E.L., Bjork, R.A., and Anderson, M.C. (1998). Varieties of goal-directed forgetting. In J.M. Golding and C.M. MacLeod (eds.), *Intentional forgetting: Interdisciplinary approaches*, volume 103. Lawrence Erlbaum, Mahwah, NJ.

Cropper, A. and Muggleton, S.H. (2016). Learning higher-order logic programs through abstraction and invention. In S. Kambhampati (ed.), *Proceedings of the 25th International Joint Conference on Artificial Intelligence*, 1418–1424.

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

Forbus, K.D. and Hinrichs, T.R. (2006). Companion cognitive systems: A step toward human-level AI. *AI Magazine*, 27(2), 83–95.

Linton, F. and Schaefer, H.P. (2000). Recommender systems for learning: Building user and expert models through long-term observation of application use. *User Modeling and User-Adapted Interaction*, 10(2-3), 181–208.

Muggleton, S. and De Raedt, L. (1994). Inductive logic programming: Theory and methods. *Journal of Logic Programming*, 19–20(Special Issue on 10 Years of Logic Programming), 629–679.

Niessen, C., Göbel, K., Siebers, M., and Schmid, U. (2017). Intentionales Vergessen im Arbeitsalltag: Eine Critical Incident Untersuchung. In *10. Tagung der Fachgruppe Arbeits-, Organisations- und Wirtschaftspsychologie der Deutschen Gesellschaft für Psychologie (AOW 2017)*. Dresden, Germany.

Niessen, C. and Jimmieson, N.L. (2016). Threat of resource loss: The role of self-regulation in adaptive task performance. *Journal of Applied Psychology*, 101(3), 450–462.

Ricci, F., Rokach, L., and Shapira, B. (2011). Introduction to recommender systems handbook. In F. Ricci, L. Rokach, B. Shapira, and P.B. Kantor (eds.), *Recommender Systems Handbook*, 1–35. Springer US, Boston, MA.

Schmid, U. (2008). Cognition and AI. *Künstliche Intelligenz*, 22(1), 5–7.

Schmid, U. and Kitzelmann, E. (2011). Inductive rule learning on the knowledge level. *Cognitive Systems Research*, 12(3), 237–248.

Schmid, U., Zeller, C., Besold, T., Tamaddoni-Nezhad, A., and Muggleton, S. (2017). How does predicate invention affect human comprehensibility? In J. Cussens and A. Russo (eds.), *Inductive Logic Programming: 26th International Conference, Revised Selected Papers*, 52–67. Springer International Publishing, Cham, Switzerland.

Yujian, L. and Bo, L. (2007). A normalized Levenshtein distance metric. *IEEE transactions on pattern analysis and machine intelligence*, 29(6), 1091–1095.