Recommendation functionality for smart data analytics toolbox to support choosing task-relevant data analytics tools

Abou Eddahab, F.; Horvath, I.

Publication date
2020

Document Version
Proof

Published in
Tools and Methods of Competitive Engineering

Proceedings of TMCE 2020, 11-15 May, 2020, Dublin, Ireland, edited by I. Horváth and G. Keenaghan

Recommendation functionality for a smart data analytics toolbox to support choosing task-relevant data analytics tools

Fatima-Zahra Abou Eddahab
Department of Design Engineering
Delft University of Technology
f.aboueddahab@tudelft.nl

Imre Horváth
Department of Design Engineering
Delft University of Technology
i.horvath@tudelft.nl

ABSTRACT

Though many enhancements are still possible and needed, data analytics software packages invaded all segments of industrial businesses. Since product designers are not specialized data analysts, an opportunity of enhancement is to provide advice by smart data analytics toolboxes (SDATBs). For instance, SDATBs can provide guidance at selecting commercially available data analytics tools (DATs) for a specific design-related task. The reported work focused on the implementation of a recommendation functionality for selecting DATs for different applications. The paper presents the proposed solution, which (i) interprets the designer's input, (ii) proposes a description of the problem identified by the designer, (iii) reasons with the warehoused DATs and (iv) recommends DATs matching the designer's task at hand. Besides presenting the needed functionality, the rules used for selecting DATs are discussed and the computational algorithms are specified. A computational feasibility testing of the tool recommendation functionality has been done considering the application case of enhancing a washing machine by white goods designers. The testing process showed that the realized functionality works correctly from a computational point of view and that it achieves sufficiently good tool matching. It compensates for the knowledge lack of product designers concerning selection of data analytics tools and reduces time and effort for tools selection. The outcomes of this study will be used in a follow up research to develop a SDATB providing even more comprehensive support for product designers.

KEYWORDS

Smart data analytics toolbox, task-relevant recommendation, machine learning tools, data analytics tools, product enhancement, white goods, support for designers.

1. INTRODUCTION

1.1. Setting the stage

Part of integrated software packages, data analytics tools (DATs) are used in multiples fields such as business, engineering, information technology, environmental studies, information systems and health informatics (Kalaian et al., 2019). Among others, they are used to extract valuable information and knowledge from huge amounts of data generated by products (Naaz & Siddiqui, 2019). Once the tools have sophisticated functionalities, they facilitate information integration and provide powerful insights to meet costumers needs and future market trends (Wang et al., 2018). In this decade, DATs have become a critical component to support decision-making processes in firms (Ghasemaghaei et al., 2018).

However, as sophistication and complexity of the tools increase, their application by non-experts is becoming less intuitive. They also become more programming intensive and need the know-how and background knowledge of data analysts and experts to get properly and efficiently used (Adhikari et al., 2018). Otherwise they diverge from their main objectives: to be transparent for all kinds of users and to be able to rapidly analyze data (Fleckenstein & Fellows, 2018). It has been observed by many researchers that the current DATs are hard to use by none data experts because they require high skill level and high efforts (Jin et al., 2017). In addition, designing smart products in a smart manner also raises new expectations towards integrated data analytics software packages.

Supporting designers and design processes of smart

products in a smart manner needs a reconsideration of both the strategy of data analytics and the functionality of the DATs. Since product designers are far from being data analysts, they need both procedural advice and decision support. Among other, these have been conceived as new functionalities for next generation data analytics software packages, called *smart data analytics toolboxes* (SDATBs). They need a reconsideration and functional adaptation of the purpose and process of using data analytics tools and they have to keep up with the era of smart products.

There are many new application contexts and functionalities foreseen. For instance, since product designers are far from being data analysts, their task completion can be supported by providing advices by a SDATB. A specific context considered in the paper, is to provide guidance at selecting data analytics tools (DATs) for a specific design-related task from the arsenal of commercially available tools. Due to the dynamic developments and the wide variety of the machine and deep learning tools, non-data scientist product designers strongly need this support (Ghoneim, 2018).

The several challenges presented by data analytics tools, especially when it comes to the satisfaction of product designers' needs, was investigated in one of our previous studies (Abouedahab & Horváth, 2018). Specifically, this study investigated the needs, dissatisfactions and expectations of designers of white goods regarding a next generation of DATs. The major findings were that white goods designers miss: (i) advising concerning applicability of data analytics tools and assistance in using them, (ii) support of acquiring and combining data from multiple data sources, (iii) combined and complementary use of qualitative and quantitative input data and (iv) means for fusing the output data of various commercialized tools.

To check the consistency of what was reflected by this investigation, the principles of axiomatic theory fusion approach were used to combine five composite theories (Horváth, 2019). These theories dealt with (i) the explicit needs of designers, (ii) the issues of interoperability, principles of decision-making, (iii) the evolution of data analytics and (iv) the evolution of enabling technologies and (v) the product design process. They proved to be useful constituents to build a robust and comprehensive conceptual basis for a knowledge platform for a SDATB. The results of the theory fusion study were in line with the findings of the first study. This confirmed our beliefs that there is a need to integrate support/help functionalities into SDATBs for designers, which assist them in their choices make them able to interpret their data and select the most relevant tools and generate outputs proper in the given design context.

1.2. Research interest and challenges

The studied literature mainly focused on (i) big data handling and its challenges such as data capturing, curation, storage and data processing performances (Srinivasan & Kumari, 2018), or (ii) big data challenges related to interoperability issues, usability and programming using analytics frameworks (Elshawi et al., 2018). The investigation related to data analytics tools concerned either (i) comparison and ranking of tools from the points of view of performance and usage (Imran et al., 2018), or (ii) providing a goal-driven description (Saggi & Jain, 2018) and analysis of the offered functionality (Gautam & Pandey, 2019).

Unfortunately, no work has been dedicated to the issue of using DATs by product designers in the context of product enhancement or other specific design tasks. While familiarity with data analytics processes are crucial for product designers, they still miss knowledge on how to select the adequate tools for processing different types of massive data sets (Abouedahab & Horváth, 2018). Consequently, the main assumption of our research was:

Assumption: *'Success of designers in product enhancement and innovation is negatively influenced by the lack knowledge about the most advanced DATs included in SDATBs'.*

The objective of the research was to address the associated challenge driven by the following guiding **research question**:

'What functionalities are to be included in a data analytics toolbox to help designers (white goods) choose the best tool to accomplish their task?'

The hypothesis towards an explanatory theory for the research has been formulated as follow:

Hypothesis: *'A recommendation service concerning the selection of the most relevant and effective tools will support the activities of white goods designers as well as enhance their experience with using DATs'.*

The objective of the research was to go beyond what was discussed in the literature and to deal with the

technical challenges of handling big data and data analytics tools. The conceived solution was developing a recommendation functionality that considers the context and specific tasks of designers and recommends the most adequate DATs for designers in order to aid data processing in a given context.

1.3. The contents and structure of the rest of the paper

The structuring of the paper's content reflects the procedural logic of the realized research activities. The next section presents the research design, including the reasoning model of the study, the different types of recommendations, as well as, the requirements of the realization of the recommendation functionality based on our approach. Section 3 introduces and details deeply the recommendation functionality, presenting their functional and algorithmic specifications. Section 4 is a demonstration of the recommendation in a real-life application case. It includes, the setting up, the realization and the interpretation of the testing. Finally, Section 5 discusses the results of the study, concludes about the findings and sketches up shorter-term and longer-term research activities.

2. RESEARCH DESIGN

2.1. Reasoning model of the study

The research addressed in this paper was framed methodologically as a design inclusive research (DIR) (Horváth, 2007). It was organized in three phases (i) explorative, (ii) constructive and (iii) confirmative. The explorative phase started with determining the research objective based on a detailed analysis of the researched phenomenon and the observed knowledge gap, driven by our research interest. The objective was used to generate the idea and the requirement for the recommendation generation. As results, the fundamental concept of the functionality was specified. The outputs of this phase were used in the constructive phase, were the concept was generated, the functional decomposition established and the high level architecture built. These elements were used in defining computational techniques and algorithms to be implemented. The confirmative phase focused on testing the feasibility and the of all computational constructs and validate their results. In this phase an application case of a washing machine enhancement by white goods designers was used.

2.2. Categories of recommendation functionalities

Based on its smart recommendation functionality, a recommendation system typically generates and provides personalized suggestions to users from a large space of alternatives or items (Srivastava & Roy, 2018). There are two main types of recommendation: (i) content-based filtering (Figure 1) and (ii) collaborative filtering (Figure 2). The first type considers the previous preferences of the user and learns a preference model using feature-based representation of the content of recommendable items. The second type of recommendation identifies preference patterns in the community of the user (Lops et

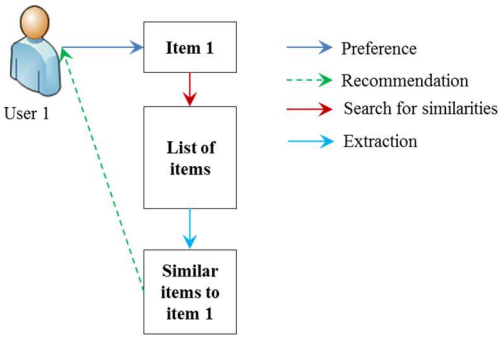


Figure 1: Content-based filtering recommendation

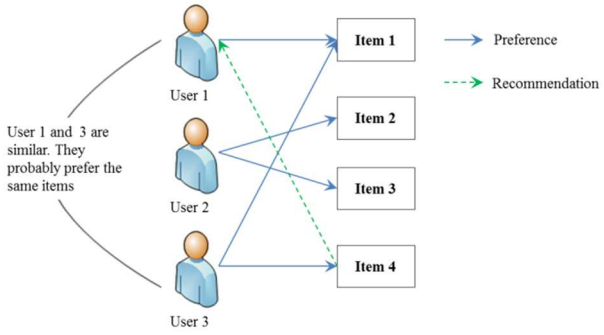


Figure 2: Collaborative filtering recommendation

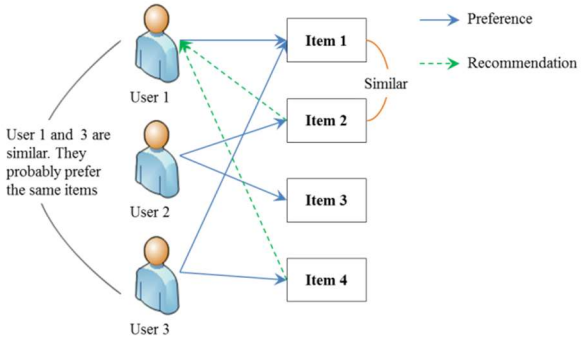


Figure 3: Hybrid recommendation

al., 2019). Another type of recommendation called hybrid recommendation was also found in the literature (Figure 3). It is a combination of content-based filtering and collaborative filtering approaches. Its objective is to attempt overcoming the short come of both approaches (Fu et al., 2018).

The objective of the recommendation in this research is to help product designers choose the best tool for processing data based on a specific design task from a set of data analytics tools. Consequently, the recommendation approach consists of recommending an item (s) (in term of DATs) to a user (designer) for whom the preferences are unknown. Based on provided user input (design task), a filtering is applied to offer a ranked and limited set of tools that are proper for purpose and can meet user's expectations. If two users provide the same input, a collaborative filtering is applied to provide similar best matching tools (Figure 4).

The adopted approach is further detailed in the upcoming sections of the paper and converted into functional specification, architecture, computational algorithms and rules. Going through these steps was challenging, not only for the novelty of the topic in product design context, but also for studying the computational feasibility and coding of the functionality.

2.3. Requirements for developing the recommendation functionality

Building a recommendation system is a task for software engineering. Requirements engineering is

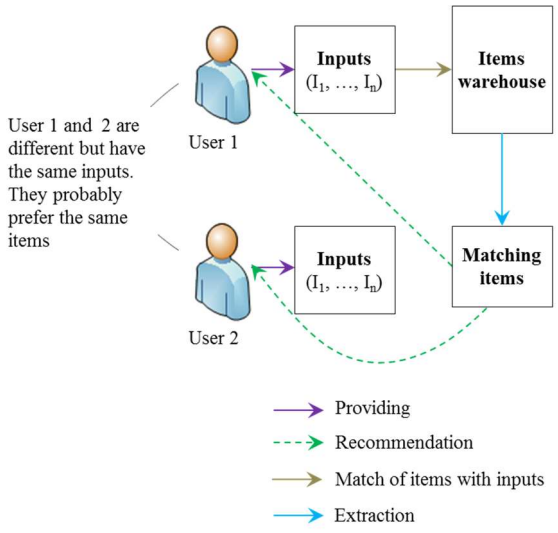


Figure 4: Recommendation for this research

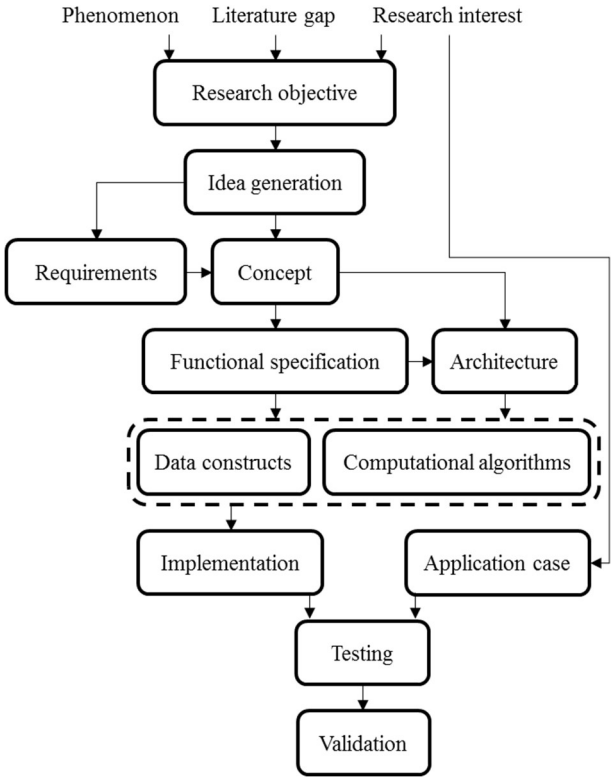


Figure 5: Reasoning model of the research

known to be one of the most critical phases in software development projects (Felfernig et al., 2013). If the requirements are poorly defined, the project often fails (Hofmann & Lehner, 2001). To define the requirements for the discussed project, a detailed investigation of the literature of existing recommendation systems and their requirements has been done (Ricci et al., 2011) (Felfering et al., 2014) (Bouzekri et al., 2017) (Li, 2019).

The list of requirements was compiled so as to guide the specification of the functionality that is supposed to be provided in our specific context. The requirements have been organized into three categories, namely (i) basic requirements (BR_x), (ii) requirements supporting designers' needs (DR_x) and (ii) technical requirements (TR_x). All categories are detailed below:

- BR_1: Computation of the recommendation functionality should be robust and in the context of data analytics tools
- BR_2: The recommendation functionality should understand the input (design task) received by the designer
- BR_3: The recommendation functionality should semantically interpret the input received by the designer

- BR_4: The computational mechanism for the recommendation functionality should be linearly computable
- DR_1: Find some / all items (DATs)
- DR_2: Find a recommended sequence of items
- DR_3: Browse recommended items
- DR_4: Annotate in context the considered item
- DR_5: Reduce time for choosing item
- DR_6: Improve designer's experience
- TR_1: Level of recommendation functionality must be identified
- TR_2: Development methods used for the recommendation must be adequate with the level of recommendation
- TR_3: Tasks performed by the designer should be explicitly described
- TR_4: Presentation and interaction with the recommendation functionality must not be error prone
- TR_5: Behavior of the functionality should be described

The established requirements will be used in all the remaining sections of this paper.

3. DETAILING OF THE RECOMMENDATION FUNCTIONALITY

3.1. Introduction of the recommendation functionality

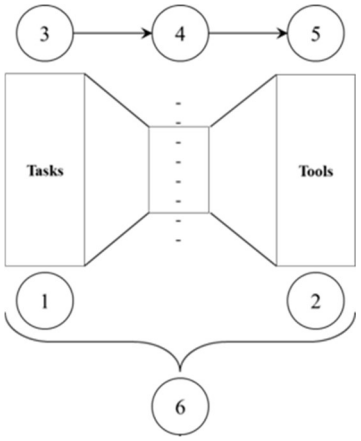
The specific objective of the recommendation functionality is to recommend the most adequate DATs for data processing in a given task context for designers. This way, time and efforts associated with tool selection will be reduced. It is assumed that recommendation functionality (i) offers a semantic interpretation of designer's input, (ii) proposes a description of the problem identified by the designer, (iii) reasons with a wide range of DATs and (iv) recommends matching tools to use based on designer's task.

For the realization of this recommendation, inputs, outputs, principles and steps of the functionality are to be defined. As inputs, the functionality requires designers' tasks (DT_x). These DT_x are to be identified and semantically analyzed to be matched with a set of DAT_x included in a warehouse of tools. The matching tools are to be ranked and the most serving one for designer's purpose is to be displayed to designer. Figure 6 shows the general idea underpinning the implementation of the recommendation functionality.

At this level some questions are to be answered to determine principles of the recommendation functionality: *'If we suppose that we have sets of DT_x and DAT_x, how to match them? And if we are able to match them, how to select the best DATs to recommend for a specific DT?'*

To answer these questions, two categories of selection criteria have been specified. For the first question, it was assumed that matching required the identification of similar features between tasks and tools. In this case, the common criteria between them need to be about "what is processed?". So, three criteria have been considered: (i) data source of the dataset to be analyzed (DS), (ii) data category of the dataset (DC) and (iii) output of data analytics (DO). These element help creating a matching matrix between DAT_x and DTx.

Concerning the ranking or the choice of the best tool, the focus in only on DAT_x and their performances. The criteria are about "how it is processed?". So, three criteria have been identified (i) graphic capabilities (C_1), (ii) speed of computation (C_2) and (iii)



(1) DT_x
(2) DAT_x warehouse
(3) DT_x identification
(4) Characterization of DT_x and DAT_x
(5) Matching DT_x and DAT_x
(6) Selection of probable DAT_x is offered

Figure 6: Concept of recommendation functionality

computational performance (C_3). A matrix of weights (W) has been allocated to all DAT based on their C_x to help ranking tools based on best/worst performances and propose to the designer the ones with the best computational performances. The (transpose) vector of weights is represented as: $W^*(WC_1, WC_2, WC_3)$.

In the next paragraph these elements and the logic behind them will be used to determine the functional specification of the recommendation functionality.

3.2. Functional specification of the recommendation functionality

The first step towards the functional specification of the recommendation functionality is the explicit specification of (i) the recommendation functionality (RF), (ii) designers input for RF (DI) and (iii) the expected output (O).

$$\begin{cases} \boldsymbol{RF}: \text{Recommendation for probable data analytics tools} \\ \boldsymbol{DI}: Designer\ task\ written\ by\ him/her \\ \boldsymbol{O}: Finite\ number\ of\ DATs\ recommended\ to\ designer \end{cases}$$

The second step is to determine the pre-established fixed elements included in the database of the DATB needed to achieve RF. These elements are the global inputs (I_x) for the functionality realization (system inputs) and they are:

$$\begin{cases} Set\ of\ DT_x \\ Set\ of\ DST_x \\ Library\ of\ DAT_x \\ Descriptions\ of\ DAT_x \\ Descriptions\ of\ DST_x \\ Matrix\ of\ Weights\ of\ DAT_x \end{cases}$$

DST_x presented above is a set of designers sub-tasks. It was added for the reason that task provided by a designer might be incomplete, insufficient or not concrete enough. So, it was decided to add DST to have bigger chances to cover all designer's tasks. This means that every DT includes a set of DSTs.

Before going deep into the decomposition of the functions for the recommendation. Let's get familiar with the steps towards the recommendation (the high level functions). These steps are shown in Figure 7.

The objective of a functional decomposition is to break down a system (in this case the recommendation functionality), starting with its main function and carry on with the intermediate levels functions up to the level of elementary functions (Erden, 1998). As the main function is known and the intermediate level functions are determined. Now, there are going to be used to obtain the lower level functions. The decomposing is explained below.

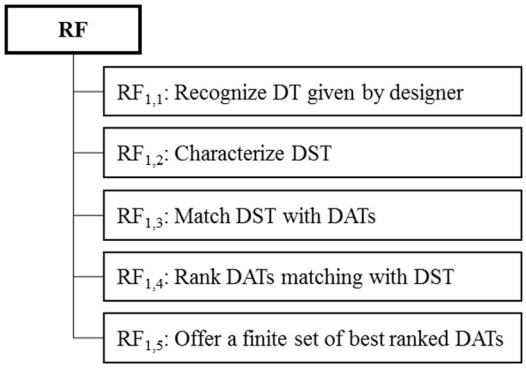


Figure 7: High level decomposition of RF

In order the system be able to recognize a DT specified by a designer ($RF_{1,1}$), it should execute a number of few steps. When the designer specifies his task using his own words, the system should be able to compare this specification of the DT with already stored DTs and DSTs, by calculating the syntactic distance between designer's inputs and the stored DTs and DSTs, and to select and retrieve the ones with the minimum distance. Since DSTs is more specific and provides more details than DTs, then the output of this sub-function is to retrieve the closest DSTs to DT written by designer. This output is used as input for the second sub-function dedicated to characterization of the chosen DST ($RF_{1,2}$). As mentioned above, a DST is characterized by DS, DC and DO.

The objective of the mentioned sub-function is to determine all these characteristics for DST and to build a vector using these pieces of information, so as DST = [DS, DC, DO]*. This vector is used as input for the third sub-function to match DST with DAT ($RF_{1,3}$). To do so, the same vector needs to be built for all DAT. Afterwards, they are used to calculate the distance between DST vector and all DAT vectors. The tools retrieved are the ones with minimum distance to DST. In this step, the weights of the selected tools are retrieved to be used together with the tools as inputs for the fourth sub-function ($RF_{1,4}$) to rank DATs based on their Ws.

The sum of weights for C_1, C_2 and C_3 of every tool has to be calculated, then sorted from high to low value. This ranks the tools from the high matching to low matching with DTs. The DATs ranking is used as input for the fifth and last sub-function

($RF_{1,5}$) that offers a finite list of ranked tools. To achieve this output, the tools with maximum sum value are selected and a final matrix of best matching DATs is produced. Figure 8 shows the low level of the functional decomposition of RF.

3.3. Architectural design of the recommendation mechanism

The RF as mentioned earlier in the paper is part of a software package for data analytics. The whole system is a next generation SDATB made for the purpose of satisfying white goods designers' needs. It

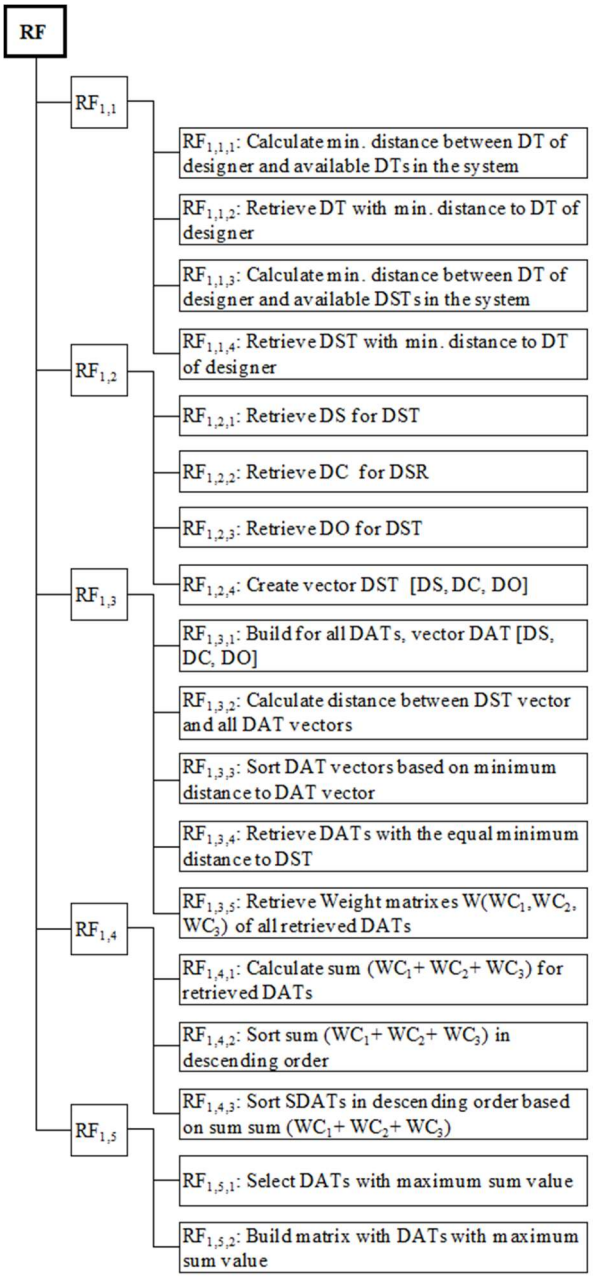


Figure 8: Low level decomposition of RF

facilitates their processing of data and supports them achieving their goal of enhancing white goods products based on product data. So, the RF is one module of the SDATB. In the architecture presented in this section, only the RF module is considered. Also, since it is the architecture of the module itself, the interactions between the system and the user are not detailed.

In section 3.2, the global inputs for the accomplishment of RF were discussed. As it can be seen, they are not all of the same nature. Correspondingly, while they are stored in the same database they require different manager units to interact with them. The DTs and DSTs are of the same nature, so one manager is able to deal with both. On the other hand, data analytics tools need a separate manager. The architecture includes the system user interface from where the DT written by designer is gotten and where the finite set of DAT is sent back to the designer. It is the way for the designer to communicate with the SDATB. Also, the database is represented. It includes sets of all global inputs.

The architecture of the RF module is shown in Figure 9. First, the design task specified by the designer is transferred from the interface to the design task explorer for identification. The written task goes though similarity calculator that uses a search engine to compare it with DTs included in the database. When the matching happens, the task will be sent to the interface going through a query task manager for interpretation of the task. The same procedure will happen for the DST identification. A sub-task descriptor is used to characterize DSTs to be used in the tools selection. This characterization is taken to the DATs explorer for matching. The explorer will search for tools in the database via the search engine and send them to the tools descriptor to look for adequate tools to DSTS. The results are sent to the recommender agent to choose the best DATs to propose. This output is then interpreted via a query manager for DATs and the final best matching data analytics tools are sent to the system user interface to be seen by the designer.

3.4. Specification of the algorithms

The realization of RF requires a number of computational rules algorithms. The functional specification and the architecture of the functionality showing how all components of the module are coming together is the entry point to understand what is expected from the computation and according to what

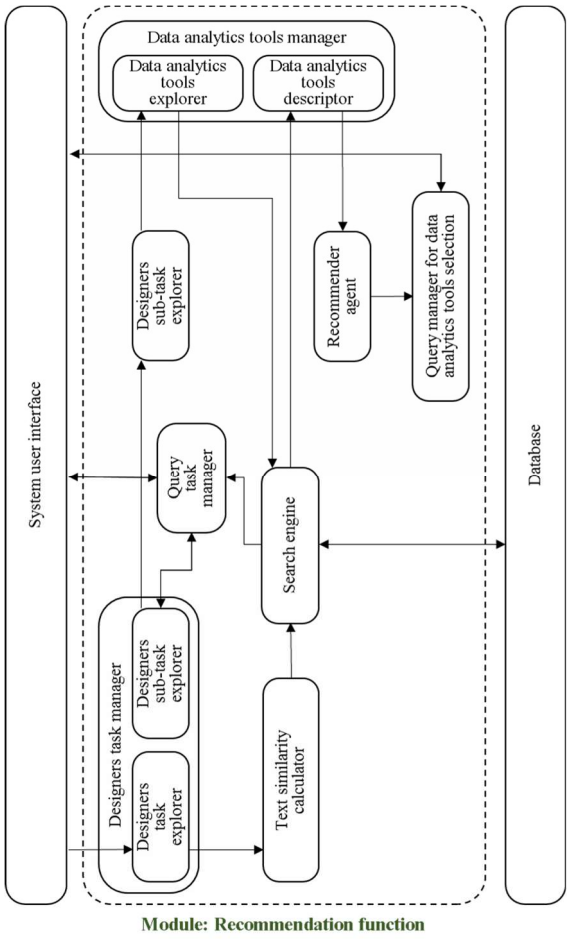


Figure 9: Architecture of the recommendation system

logic and order. All algorithms are written with the resources of the Matlab software package.

To realize the function $RF_{1,1}$, a function calculating the minimum distance between DT of designer and available DTs in the system needs to be defined. This function is known as "EditDistance". It is a standard dynamic programming problem. Given two strings s1 and s2 (e.g., words, sentences), the EditDistance between s1 and s2 is the minimum number of operations required to convert string s1 to s2. Using Matlab, the edit distance is defined as follows:

```
function [V,v] = EditDistance(string1,string2)
    m = length(string1);
    n = length(string2);
    v = zeros(m+1,n+1);
    for i = 1:1:m
        v(i+1,1) = i;
    end
    for j = 1:1:n
        v(1,j+1) = j;
    end
    for i = 1:m
        for j = 1:n
            if (string1(i) == string2(j))
                v(i+1,j+1) = v(i,j);
            else
                v(i+1,j+1) = 1+min(min(v(i+1,j),
                v(i,j+1)), v(i,j));
            end
        end
    end
    V = v(m+1,n+1);
end
```

In the case of RF, s1 is the DT given by designer and instead of s2, all DTs saved in the system will be considered. This can be done by define-ng a character array (char), that is a sequence of vectors (textual, numerical). The objective is not to calculate EditDistance, but its minimum between DT of designer and char of DTs known to the system (general input number 1: I_1) and DST (I_2). The output is the DST closest to designer's DT. To implement this functionality, Algorithm 1 was implemented as shown above.

The objective of $RF_{1,2}$ is to characterize obtained DST based on DS, DC and DO. In order to deter-

Algorithm 1. Retrieve DST with minimum distance to DT given by designer

Inputs: I1 = {DT_1, DT_2, ..., DT_n}
I2 = {DST_1, DST_2, ..., DST_n}
I3 = DT_x written by designer

Outputs: DST_x closest to I_3

```
DTX = ' I3 ';
str = char(I2(1));
s = 1000;
aux = str;
for idx1 = 1: length(I2)
    str = char (I2(idx1));
    if (EditDistance (DTX, str)<s)
        s = EditDistance (DTX, str);
        aux = str;
        IND = idx 1;
    end
end
if (IND < numerical value 1)
    DTX = DT1;
elseif (IND < numerical value 2)
    DTX = DT2;
     (continue until DTn-1)
else
    DTX = DTn;
end
```

mine these characteristics, patterns have to be recognized. For example, if the expression of DST_x retrieved starts with the word "analyzing" then DST_x can have one of two alternative characterizations, $DC_x = DC_1$ and $O_x = [O_1, O_2]$, else $DC_x = DC_2$ and $O_x = [O_6, O_8, O_{10}]$. For this sub-function, an implicit transformation matrix is used, which enables a slightly more complicated pattern recognition for DS_x. The procedure consists of converting the textual description of DSTs to a vector of words, then compare it with constant character vectors representing relationships between DST_x and DS_x. Two functions are needed for this matter, (i) "strsplit" to split strings (DST_x) and (ii) "strcmp" to compare them. The output of $RF_{1,2}$ is a DST vector including DS, DC and O. Algorithm 2 was implemented to achieve the output. The mentioned example is used to simplify the writing of the algorithm.

Algorithm 2. Build DSTs vectors

Inputs: $I4 = \{DS_1, DS_2, ..., DS_n\}$
$I5 = \{DC_1, DC_2, ..., DC_n\}$
$I7 = \{DAT_1, DAT_2, ..., DAT_n\}$
Outputs: DST_X vector $[DS_x, DC_x, O_x]$

```
1:  DSTXVEC = strsplit(DSTX);
2:  if (strcmp (DSTXVEC(1), ' word 2 '))
3:      DCX = DC1;
4:      OX = {O1, O2};
5:  else
6:      DCX = DC2;
7:      OX = {O6, O8, O10};
8:  end
9:  for idx1 = 2 : length (DSTXVEC)
10:     if (strcmp (DSTXVEC(idx1), ' word 3 ' == 1 )
11:         DSX = DS1;
12:     end
13:     if (strcmp (DSTXVEC(idx1), ' word 4 ' == 1 )
14:         DSX = DS2;
15:     end
10:     if (strcmp (DSTXVEC(idx1), ' word n ' == 1 )
11:         DSX = DSn;
12:     end
13: end
```

To realize $RF_{1,3}$, first a cell array VectorTools need to be built for all DATs as shown in Algorithm 3. To better illustrate the logic of the algorithm flow and functioning, examples of possible inputs characterization have been used (e.g. Line 3 of Algorithm 3).

Algorithm 3. Build vector DAT

Inputs: $I4 = \{DS_1, DS_2, ..., DS_n\}$
$I5 = \{DC_1, DC_2, ..., DC_n\}$
$I6 = \{O_1, O_2, ..., O_n\}$
$I7 = \{ DAT_1, DAT_2, ..., DAT_n \}$
Outputs: DAT_X vector $[DS_x, DC_x, O_x]$

```
1:  for idx1 = 1 : length (I7)
2:      if (idx1 == 1 | | idx == 4| | idx1 == 8)
3:          DAT = [DS2  DS4  DC1];
4:      else
5:          DAT = [DS1  DS3  DS5  DS6  DC1  DC2];
6:      end
7:      if (idx == 1)
8:          DAT = [DAT  O5];
9:      else if (idx==4)
10:         DAT = [DAT  O4];
11:     else
12:         DAT = [DAT  O1];
13:     end
14:     VectorTools {idx1} = DAT
15: end
```

As a second step, Algorithm 4 calculates the distance between vectors DST and DAT. To do so, two functions introduced (i) "strfind" determines patterns within strings and (ii) "isempty" checks whether a string is empty or not.

Algorithm 4. Calculate distance between DSTs and DAT vectors

Inputs: no new inputs
Outputs: distance between DSTX and DATX

```
1:  for idx = 1 : length (VectorTools)
2:      distance = 0;
3:      for idx2 = 1: length (DSTVECTOR)
            distance = distance + isempty (strfind (char
              (VectorTools (idx1)), DSTVECTOR
              (idx2)));
4:      end
5:      distanceVector(idx1) = distance;
6:  end
```

The third step is to sort DATs included in I_7 according to their distance to DAT. This is done by Algorithm 5, shown below. The computational essence of the used algorithm is "bubble sort", also known as sinking sort. It is a sorting algorithm that steps through a list, compares adjacent elements and shift them if they are in the wrong order. Its logical specification is as follows:

Algorithm 5. Sorting DATs

Inputs: $I7 = \{ DAT_1, DAT_2, ..., DAT_n \}$
Outputs: DATX sorted by distance

```
1:  for idx = 1 : length (I7) : -1 : 1
2:      for idx2 = 2 : idx1
3:          if (distanceVector (idx2-1) > distanceVector
              (idx2))
4:              tmp = I7 (idx2-1);
5:              I7 (idx2-1) = I7(idx2);
```

```
6:              I7(idx2) = tmp;
7:              tmpd = distanceVector(idx2-1);
8:              distanceVector (idx2-1) = distanceVec-
                  tor(idx2);
9:              distanceVector (idx2) = tmpd;
10:           end
11:       end
12:   end
```

Once the distance is calculated, all DATs that have equal minimum distance (the most similar vectors to DST) are retrieved. The tools weights (I_8) are also retrieved to be used later on. The computational details can be seen in the specification of Algorithm 6.

Algorithm 6. Retrieve most similar DAT vectors to DST vector

Inputs: $I8 = \{W_1, W_2, ..., W_n\}$
$I7 = \{DAT_1, DAT_2, ..., DAT_n\}$
Outputs: DATs with equal minimum distance to DST

```
1:  distance = distanceVector (1);
2:  DATs = [I7 (1)];
3:  weightsSimilarVector = [I8 (1)];
4:  i = 2;
5:  While (distance == distanceVector (i))
6:      DATs = [DATs  I7(i)];
7:      weightsSimilarVector = [weightsSimilarVector
          I8(i)];
        i = i+1;
8:      if (i > length(distanceVector))
9:          break;
10:     end
11:   end
```

In order to rank the tools with minimum distance to DAT, the matrix of weights is used. Every tool has 3 different weights, one for each criterion (C_x). First, a vector of weights sum is generated. Then, the sum vector is sorted in a descendant order (RW: weights sorted or ranked). Correspondingly, the tools are also sorted (RT: ranked tools). The step is implemented by Algorithm 7.

The final step is to select tools of maximum sum and present the final set of the best matching tools to designer. The function used is maxSom that determines the maximum sum of weights (MW). The simple logic of the considered algorithm is presented below (Algorithm 8).

Algorithm 8. Retrieve best finite set of DATs

Inputs: no new inputs
Outputs: Matrix of DATs with high weights

```
1:  MW = max(somWC1(1));
2:  FinalMatrixTi = [];
3:  for i = 1: length(MW)
```

Algorithm 7. Ranking DATs

Inputs: $I8 = \{W_1, W_2, ..., W_n\}$
$I7 = \{DAT_1, DAT_2, ..., DAT_n\}$
Outputs: DATs sorted from high to low

```
1:  for idx1 = 1 : length (weightsSimilarVector)
2:      W = char(weightsSimilarVector(idx));
3:      WC1 = W(1);
4:      WC2 = W(2);
5:      WC3 = W(3);
6:      somWC1 (idx1) = WC1 + WC2 + WC3;
7:  end
8:  RW = sort(somWC1, 'descend');
9:  [c,d] = sort (somW1, 'descend');
10: RT = [];
11: for i = 1: length(somWC1)
12:             RT = [RT, Tis(d(i))];
13: end
```

```
4:      FinalMatrixTi(i) = [FinalMatrixTi, Tis(d(i))];
5:  end
```

Considering the specification of the algorithms and the presented architecture of the recommendation, the relationship between algorithms was established. Figure 10 shows the communication between algorithms, data exchanges and conditions. The elements in green are the data inputs and outputs of each algorithm. It can be seen in the figure that some boxes do not belong to any algorithm. These are to be determined when studying the type of interactions system/system and system/designer, mentioned in the architecture as query managers and system user interface. In this paper and in the demonstration presented in the next section, the focus is on evaluating the feasibility of RF from a computational point of view. For this reason, interactions and interface development are not presented.

4. A DEMONSTRATIVE APPLICATION

4.1. Activation of the SDATB recommendation function

The goal of this demonstrative application is to activate the recommendation function (included in a next generation SDATB) in the case of a concrete application. For testing the feasibility of RF, we decided to target an application related to white goods. This interest came from the fact that white goods cover a large field of product. They are heavy consumer durables that include all home appliances related to refrigeration, cooking, washing and drying equipment, as well as, heating and cooling. Also, these products are equipped with advanced control units and a relatively high number of sensors able to

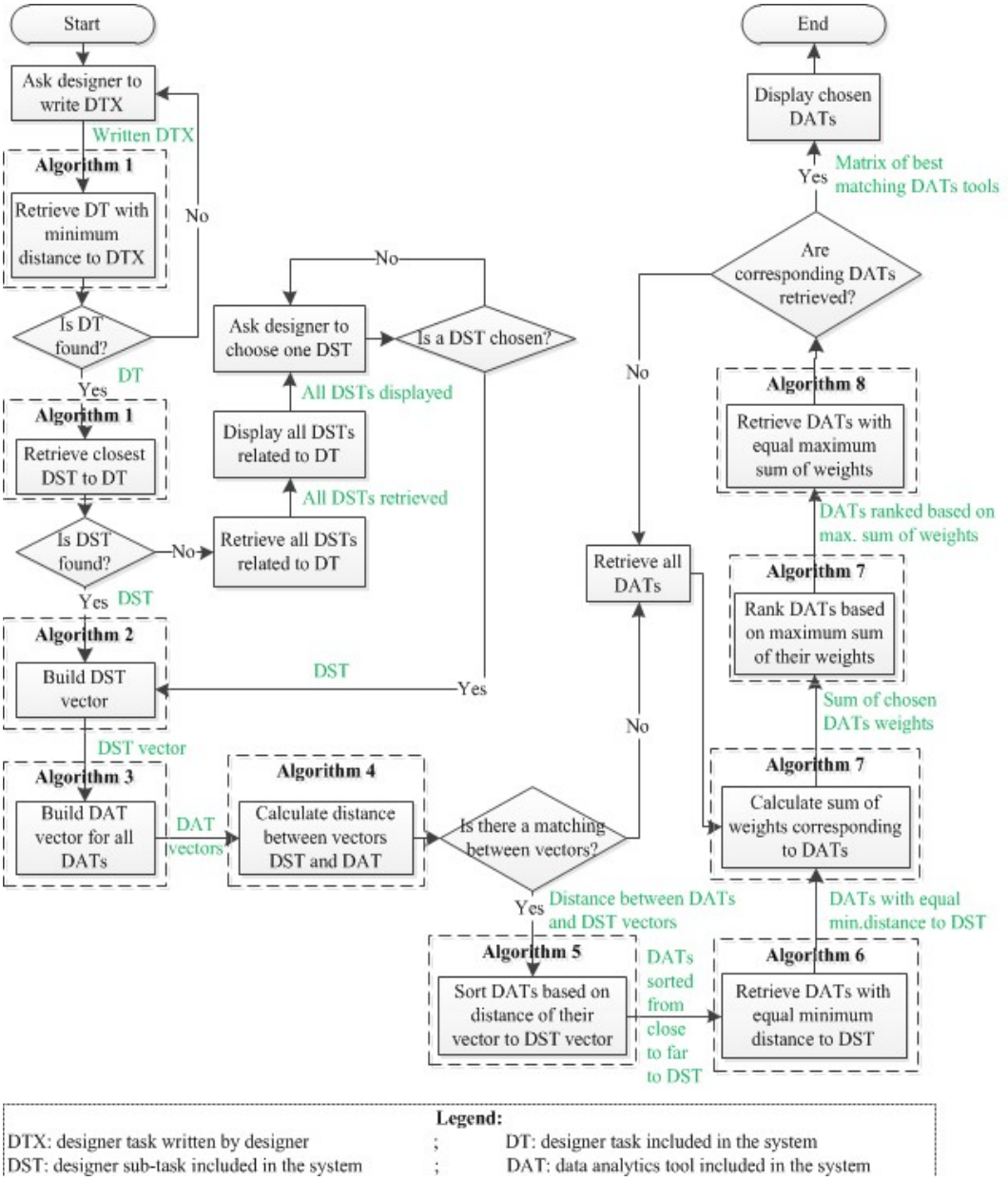


Figure 10: Logical flowchart of the recommendation functionality

collected product data. The majority of them is known by the continues evolution towards smart products. In this sense, the agreement was to use the RF to recommend the appropriate DATs for designers in the process of enhancement of a particular connected washing machine using SDATB.

Before jumping into the testing of RF, the global inputs have to be specified. The first step is the identification of DTs and their related DSTs as I_1 and I_2. For the sake of simplification, three DT_x are selected and DST will be represented as $DT_{x,y}$ (where x is the code of the main DT and y the order of the DST):

- DT_1: Enhancement of product performance:
  - $DT_{1,1}$: Analyzing energy consumption
  - $DT_{1,2}$: Analyzing water consumption
  - $DT_{1,3}$: Analyzing temperature settings
  - $DT_{1,4}$: Analyzing loading
  - $DT_{1,5}$: Analyzing detergent usage
- DT_2: Enhancement of product design:
  - $DT_{2,1}$: Analyzing most used features
  - $DT_{2,2}$: Analyzing relationships between most used features
  - $DT_{2,3}$: Analyzing least used features
- DT_3: Enhancement of product life cycle:
  - $DT_{3,1}$: Analyzing product components

- $DT_{3,2}$: Scheduling of predictive maintenance
- $DT_{3,3}$: Scheduling of preventive maintenance

The second step is the identification of DS of the washing machine. Seven data sources have been identified:

- DS_1: Temperature sensor (for $DT_{1,3}$)
- DS_2: Water flow sensor (for $DT_{1,2}$)
- DS_3: Load sensor (for $DT_{1,4}$)
- DS_4: Detergent level sensor (for $DT_{1,1}$, $DT_{1,5}$)
- DS_5: Event log (for $DT_{2,1}$, $DT_{2,2}$, $DT_{2,3}$, $DT_{3,1}$)
- DS_6: Maintenance history (for $DT_{3,2}$)
- DS_7: Maintenance report (for $DT_{3,3}$)

The third step is the identification of DC. For simplification, two main categories of data were identified:

- DC_1: Big data (coming from DS_1, DS_2, DS_3, DS_4, DS_5)
- DC_2: Small data (coming from DS_6, DS_7)

The fourth step is the identification of possible O_x based on data category. Since the testing is a feasibility testing, a limited number of outputs is used:

- O_1: Plots (possible for DC_1, DC_2)
- O_2: Hierarchical tree (possible for DC_1, DC_2)
- O_3: Dendrogram (possible for DC_2)
- O_4: Hyperplane (possible for DC_2)

The fifth step is the identification of DATs that are included in the SDATB (referred to as T_x, for simplification and avoiding coding errors later on). For the purpose of demonstration, some machine learning tools from the "Statistics and machine learning toolbox" of Matlab were considered, as listed below:

- T_1: Support vector machines (can analyze DC_2, and provide O_4)
- T_2: Decision trees (can analyze DC_1 and DC_2 and provide O_2)
- T_3: Classification trees (can analyze DC_1, DC_2, O_2)
- T_4: K-nearest neighbor (can analyze DC_2 and provide O_1)
- T_5: K-means (can analyze DC_1 and DC_2 and provide O_1)
- T_6: K-medoids (can analyze DC_1 and DC_2 and provide O_1)
- T_7: Hierarchical clustering (can analyze DC_2 and provide O_3)
- T_8: Gaussian mixture models (can analyze DC_2 and O_1)

The last step is the identification of weight matrix for each of the tools according to C_1, C_2 and C_3. If our algorithm was a machine learning one, the weights can be automatically defined. To avoid fundamental mistakes, it is important to mention that the weights were arbitrary generalized for the purpose of the testing purpose of RF and how the tools with high weights value were chosen. Below is the list of eight weight matrixes for the corresponding eight tools:

- W_1 = [3 10 5];
- W_2 = [2 7 4];
- W_3 = [8 3 1];
- W_4 = [1 6 8];
- W_5 = [7 5 5];
- W_6 = [7 4 7];
- W_7 = [10 1 1];
- W_8 = [1 3 6];

Figure 6 is revisited, adapted and specialized to the application case, as shown in Figure 11. On the left side of Figure 6, abbreviations of designers tasks and sub-tasks that are detailed previously are cited. On the right side of the same figure, abbreviations of data analytics tools for testing are listed. In the middle of the figure is the matching matrix between designer's tasks and tools based on DS, DC and O criteria. This figure is generated for the purpose of checking the validity of the RF. If the design task specified by the designer is closer to a specific DT_x and $DT_{x,y,}$ as shown in Figure 11, then the DATs to be recommended for the designer should be the tools most corresponding to DT_x and $DT_{x,y}$ in the list represented on the right side in Figure 11. The next step

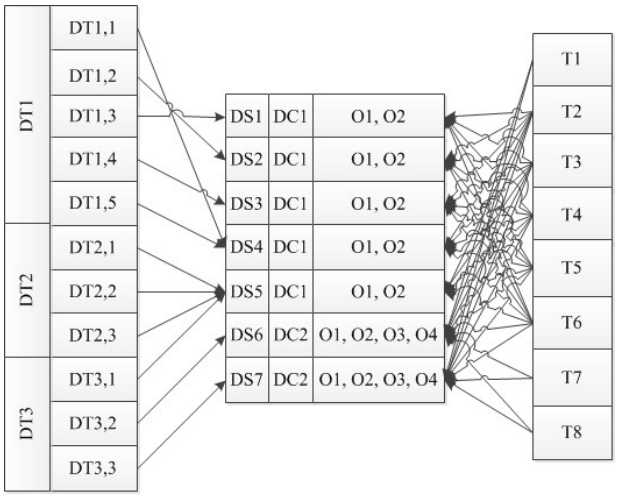


Figure 11: Adaptation of recommendation principle in a particular application case

is to test the feasibility of RF.

4.2. Computational feasibility validation by processing the algorithms

The testing of RF was completed by using Matlab. The starting point was the input of data from a white goods design exercise for which DATs will be selected. The objective was to demonstrate that RF is computationally feasible. First, the sets of RF inputs and codes corresponding to all algorithms are inserted and written in a new "Matlab script". Then, in order to be able to compile the script, the functions used in the algorithms and that are unknown to Matlab (all functions are presented in section 3.4) have to be defined (e.g. EditDistance).

The sets of inputs are defined and inserted in Matlab as:

```
%--------Designers Tasks--------
DT1 = 'Enhancement of product performance';
   DT11 = 'Analyzing energy consumption';
   DT12 = 'Analyzing water consumption';
   DT13 = 'Analyzing temperature settings';
   DT14 = 'Analyzing loading';
   DT15 = 'Analyzing detergent usage';

DT2 = 'Enhancement of product design';
   DT21 = 'Analyzing most used features';
   DT22 = 'Analyzing relationships between most
          used features';
   DT23 = 'Analyzing least used features';

DT3 = 'Enhancement of product life cycle';
   DT31 = 'Analyzing product components';
   DT32 = 'Scheduling of predictive maintenance';
   DT33 = 'Scheduling of preventive maintenance';

%--------Machine Learning tools--------
T1 = 'Support vector machines';
T2 = 'Decision trees';
T3 = 'Classification trees';
T4 = 'K-nearest neighbor' ;
T5 = 'K-means';
T6 = 'K-medoids';
T7 = 'Hierarchical clustering';
T8 = 'Gaussian mixture models';

%--------Data Sources--------
DS1 = 'Temperature sensor';
DS2 = 'Water flow sensor';
DS3 = 'Load sensor';
DS4 = 'Detergent level sensor';
DS5 = 'Event log';
DS6 = 'Maintenance history';
DS7 = 'Maintenance report';

%--------Data categories--------
DC1 = 'Big Data';
DC2 = 'Small Data';

%--------Outputs expected--------
O1 = 'Plots';
O2 = 'Hierarchical tree';
O3 = 'Dendrogram';
O4 = 'Hyperplane';

%--------Weights--------
W1 = [3 10 5];
W2 = [2 7 4];
W3 = [8 5 4];
W4 = [1 9 8];
W5 = [7 6 2];
W6 = [2 4 7];
W7 = [10 1 1];
W8 = [1 3 6];

%--------GLOBAL INPUTS--------
ISB11 = {DT1, DT2, DT3};
%set of tasks
ISB12 = {DT11, DT12, DT13, DT14, DT15,
DT21, DT22, DT23, DT31, DT32, DT33};
%set of subtasks
ISB14 = {T1, T2, T3, T4, T5, T6, T7, T8};
```

EditDistance.m | washingmachine.m

```
56      %---------------------- RF 1.1 ------------------
57 -    DTX ='washing machine load';
58 -    str=char(ISB12(1));
59 -    s=1000;
60 -    aux=str;
61 -    for idx1 = 1:length(ISB12)
62 -        str=char(ISB12(idx1));
63 -        if (EditDistance(DTX,str)<s)
64 -            s=EditDistance(DTX,str);
65 -            aux=str;
66 -            IND=idx1;
67 -        end
68 -    end
69
70 -    if(IND<6)
71 -        DTX=DT1;
72 -    elseif (IND<9)
73 -        DTX=DT2;
74 -    else
75 -        DTX=DT3;
76 -    end
```

Command Window

```
DTX =

    'Enhancement of product performance'

>> DTXY

DTXY =

    'Analyzing loading'
```

Figure 12: Process and outputs of $RF_{1,1}$

```
%set of machine learning tools
ISB110 = {W1, W2, W3, W4, W5, W6, W7, W8};
%weight matrix
```

Now, it is time for the designer to write down his design task as an entry point to RF. In this test case, we consider that the designer wants to analyze a washing machine loading. We assume that he wrote the design task without "verb" as follow: DTX = 'washing machine load'.

The pre-defined algorithms in Section 3.4 were converted into codes in the particular application case and inserted in Matlab as well as the textual formulation of DTX. Running the RF codes made us achieve the following:

$RF_{1,1}$ is represented in Figure 12. The obtained outputs are: $DT_x = DT_1$ ='Enhancement of product performance' and $DT_{x,y} = DT_{1,4}$ = 'Analyzing loading'. $RF_{1,2}$ is represented in Figure 13.

The output is a vector characterizing $DT_{1,4}$ in terms of data source, data category and outputs. These characteristics are: $DS_x = DS_3$ = load sensor, $DC_x = DC_1$ = big data, O_1 = plots and O_2 = hierarchical tree. $RF_{1,3}$ is represented in Figure 14. The output is a vector of data analytics tools matching $DT_{1,4}$. These tools are: T_2 = Decision trees, T_3 = Classification trees, T_5 = K-means, and T_6 = K-medoids.

$RF_{1,4}$ is represented in Figure 15. The outputs are the matrix of sum of weights [18, 13, 17, 12] and the ordering of the sum matrix [18, 17, 13, 12]. Finally, the ordering of tools is [T_6, T_5, T_2, T_3].

Finally, $RF_{1,5}$ is represented in Figure 16. Two outputs are expected and obtained, the maximum sum of weights (18) and the corresponding tools to offer to designer, [T_6].

```
EditDistance.m | washingmachine.m*
77  %------------------------ RF 1.2 ------------------------
78  DTXYVEC = strsplit(DTXY);
79  if (strcmp(DTXYVEC(1),'Analyzing'))
80      DCX=DC1;
81      OX = {O2,O6};
82  else
83      DCX=DC2;
84      OX = {O2,O6,O8,O10};
85  end
86  for idx1 = 2:length(DTXYVEC)
87      if(strcmp(DTXYVEC(idx1),'temperature')==1)
88          DSX=DS1;
89      end
90      if(strcmp(DTXYVEC(idx1),'water')==1)
91          DSX=DS2;
92      end
93      if(strcmp(DTXYVEC(idx1),'loading')==1)
94          DSX=DS3;
95      end
96      if(strcmp(DTXYVEC(idx1),'energy')==1||strcmp(DTXYVEC(idx1),'detergent')==1)
97          DSX=DS4;
98      end
99      if(strcmp(DTXYVEC(idx1),'features')==1||strcmp(DTXYVEC(idx1),'components')==1
100         DSX=DS5;
101     end
102     if(strcmp(DTXYVEC(idx1),'predictive'))
103         DSX=DS6;
104     end
105     if(strcmp(DTXYVEC(idx1),'preventive'))
106         DSX=DS7;
107     end
108
109 end

Command Window
  DTXYVECTOR =

    1×4 cell array

    'Load sensor'    'Big Data'    'Plots'    'Hierarchical tree'
```

Figure 13: Process and outputs of $RF_{1,2}$

```
EditDistance.m | washingmachine.m
110 %------------------------ RF 1.3 ------------------------
111 for idx1 = 1:length(ISB14)
112     if (idx1==1 || idx1==4 || idx1==7 || idx1==8)
113         T = [DS6 DS7 DC2];
114     else
115         T= [DS1 DS2 DS3 DS4 DS5 DS6 DS7 DC1 DC2];
116     end
117     if (idx1==1)
118         T=[T O10];
119     elseif (idx1==7)
120         T=[T O8];
121     elseif (idx1==2 || idx1==3)
122         T=[T O6];
123     else
124         T=[T O2];
125     end
126     VectorTools{idx1} =  T;
127 end
128 for idx1 = 1:length(VectorTools)
129     distance = 0;
130     for idx2 = 1:length(DTXYVECTOR)
131         distance = distance + isempty(strfind(char(VectorTools(idx1)),DTXYVECTOR(idx2)));
132     end
133     distanceVector(idx1) = distance;
134 end
135 for idx1 = length(ISB14):-1:1
136     for idx2 = 2:idx1
137         if (distanceVector(idx2-1)>distanceVector(idx2))
138             tmp = ISB14(idx2-1);
139             ISB14(idx2-1) = ISB14(idx2);
140             ISB14(idx2) = tmp;
141             tmpd = distanceVector(idx2-1);
142             distanceVector(idx2-1) = distanceVector(idx2);
143             distanceVector(idx2) = tmpd;
144         end
145     end
146 end
147 i=2;
148 while(distance==distanceVector(i))
149     TIs = [TIs ISB14(i)];
150     weightsSimilarVector = [weightsSimilarVector ISB110(i)];
151     i=i+1;
152     if (i>length(distanceVector))
153         break;
154     end
155 end

Command Window
  >> TIs

  TIs =

    1×4 cell array

      'K-medoids'    'Decision trees'    'K-means'    'Classification trees'
```

Figure 14: Process and outputs of $RF_{1,3}$

```
EditDistance.m | washingmachine.m
156 %------------------------ RF 1.4 ------------------------
157 for idx1 = 1:length(weightsSimilarVector)
158     W = char(weightsSimilarVector{idx1});
159     WC1 = W(1);
160     WC2 = W(2);
161     WC3 = W(3);
162     somWC1(idx1) = WC1 + WC2 + WC3;
163 end
164 RW = sort(somWC1, 'descend');
165 [c,d]=sort(somWC1, 'descend');
166 RT=[];
167 for i=1: length(somWC1)
168     RT=[RT, TIs(d(i))];
169 end

Command Window
  >> somWC1

  somWC1 =

      18    13    17    12

  >> RW

  RW =

      18    17    13    12

  >> RT

  RT =

    1×4 cell array

      'K-medoids'    'K-means'    'Decision trees'    'Classification trees'
```

Figure 15: Process and outputs of $RF_{1,4}$

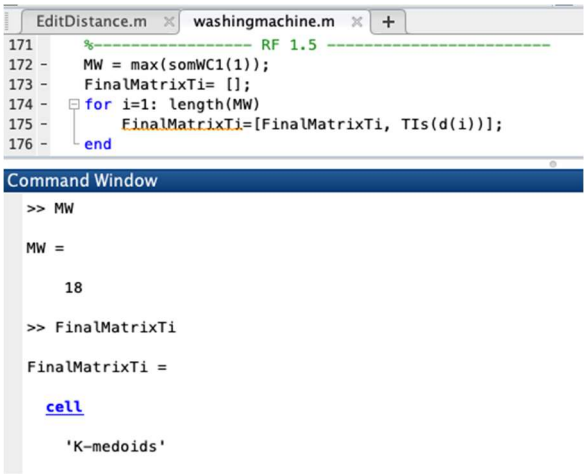


Figure 16: Process and outputs of $RF_{1,5}$

By referring to Figure 11, the tools selected based on DTX (DT_1) are [T_2, …, T_6] and the recommended one based on weight sum is [T_6] (by calculating the weights sum). This means that RF provides the best match.

4.3. Interpretation of test results

The objective of the realized testing is to check the feasibility and the functionality of the recommendation. It consists of verifying if the algorithms do what they are supposed to do and if the requirements are fulfilled. In other words, the correctness of the computation and the achievement of sub-functions are evaluated.

The results of the computational feasibility testing are shown from Figure 12 to Figure 16. By analyzing these figures from a feasibility perspective, the following points were identified: (i) the algorithms can be implemented, (ii) they are computationally correct, (iii) the codes do not contain any errors, (iv) each algorithm set of inputs and outputs is identical to the expected results presented in the algorithms description in section 3.4, (v) the algorithms communicate with each other, since the output of one algorithm is the input for the following one, also (vi) no conversions of inputs or outputs were needed all along the functionality implementation.

Based on the outcome of the computational feasibility testing, we may conclude that the flow of algorithms is coherent and that recommendation function is dependable. By analyzing the whole of the computational process, it was observed that starting from the initial input DTX given by the designer, the output was a finite matrix of DATs with high weights values. This was also confirmed in section 4.2, where RF provided the same results both computationally and manually. If the designer provides the needed sufficiently complete specification of the intended task, the tools to be used could be found (recommended) automatically. This leads to the conclusion that the computational implementation of the RF module achieves the desired output. RF is also performant, since it (i) facilitates the choice process of designers, (ii) saves time and efforts related to this matter, and (iii) compensates for the lack of knowledge of the designer regarding DATs.

Having the RF implemented as part of the next generation SDATB, semantic support can be provided for product designers. Instead of getting lost in the huge amount of DATs and their new updates, the designers will get the information about the most suitable tools in seconds, while it would take long hours without this smart functionality. The proposed FR makes it possible for them to focus on their mainstream design tasks enabling product enhancements, rather than being stuck with learning the capabilities of novel DATs when they want to get information from MoLD. They may avoid getting busy with investigating, studying and comparing DATs that might or might not be suitable for their DTX.

5. DISCUSSION, CONCLUSIONS AND FUTURE WORK OF THE RESULTS

5.1. Discussion

The objective of the research presented in this paper is the specification and the implementation of a RF for a next generation SDATB in terms of algorithms and data constructs. Based on research actions dedicated for this purpose, some limitations were recognized.

- Using machine learning algorithms might have reduced the time for code building. Since some inputs (e.g. weights, tools) can be generated and adjusted automatically. This is to be considered for future improvements of RF.
- DTX are supposed to be known for the system. Syntactic matching is being implemented. This one of the RF limitations. A semantic matching based on synonyms is to be considered for the future.
- For the testing a small range of all system inputs are considered (DTX, DATs, ...) which made the computing easy to realize. In case of wide range of

inputs, the computing might be time consuming and human mistakes are likely to happen. In the future, automation methods for inputs insertion are to be studied.

- The RF is being individually implemented which does not reflect the performance of the whole SDATB to which it belongs. RF needs to be tested in the future as part of the complete next generation SDATB.

5.2. Conclusions

The testing process and results showed that the recommendation functionality works correctly from a computational point of view. In our research, we did not test RF is a real-life environment, for the reason that it will go beyond the scope of the work that consists of offering tools into solving design problems using machine learning tools. Another aspect not considered was: what to do if the tools proposed are with low weight or even no tool is proposed? This problem means that new tools are to be included in the SDATB, which remains a knowledge engineer task.

The main conclusions of this work based on testing results are:

- RF is able to interpret designer's inputs and propose a description of the DTX identified.
- RF reasons with DATs and recommends the best matching one to DTX.
- The rule for DATs selection and recommendation is captured by their weight and the matching matrix between DTs and DATs.
- The recommended tool was proven to be the most adequate one based on DTX.
- The recommendation function compensates for knowledge deficiency of DATs by product designers in particular task.
- The recommendation function reduces time and efforts associated with tool selection.

5.3. Future research

The on-going part of the research is to improve the recommendation functionality considering the recognized limitations. The outcomes of this study will be used in a follow up research to a develop a smart data analytics toolbox providing comprehensive support for product designers.

Based on past studies a set of global set of needs and expectations of SDATB was determined. After filtering of this set based on needs and expectations covered of partially covered by the literature, a final reduced set of needs and expectations of the SDATB was kept for further investigation. This final set will be used in the conceptualization of the next generation data analytics toolbox. It includes: (i) adaptation of the toolbox to the user, (ii) semantic interpretation of the analysis outcomes, (iii) learning from toolbox applications, (iv) affording permanent accessibility of the toolbox, and (v) offering a step by step assistance all along the SDATB usage. In this sense, future research activities consist of conceptualizing, architecting and computationally realizing all functionalities to answer white goods designers needs presented in our background study. The structural and computational means for these functionalities combination will be studied. Finally, the set of functionalities will be used to build a next generation smart data analytics toolbox tailored for white goods designers' needs. The SDATB will include smart reasoning and learning mechanisms. These are needed to address meanings and semantic interpretations in the process of analyzing design tasks. This will make it easy for designer not only to analyze data but also to interpret them and make decisions about them. The smartness of SDATB will help in keeping up with the fast improvement of smart products.

REFERENCES

[1] Abou Eddahab, F.Z., & Horváth, I. (2018). What do designers miss regarding the outputs of data analytics tools in the context of possible product improvements?, In: *Proceeding of the 12th Symposium on Tools and Methods of Competitive Engineering*, 1-14.

[2] Adhikari, B.K., Zuo, W.L., Maharjan, R., & Yadav, R.K. (2018). Use of big data analytics in wash sector. In: *Proceeding of Second International Conference on Intelligent Computing and Control Systems.* IEEE, 1185-1190.

[3] Bouzekri, E., Canny, A., Fayollas, C., Martinie, C., Palanque, P. A., Barboni, E., ... & Gris, C. (2017). A list of pre-requisites to make recommender systems deployable in critical context. In: *Proceeding of EnCHIReS@EICS*, 42-55.

[4] Elshawi, R., Sakr, S., Talia, D., & Trunfio, P. (2018). Big data systems meet machine learning challenges: Towards big data science as a service. *Big Data Research*, 14, 1-11.

[5] Erden, Z., Erkmen, A.M., & Erden, A. (1998). A Petri net-based design network with applications to mechatronic systems. *Journal of Integrated Design and Process Science*, 2, 32-48.

[6] Felfernig, A., Ninaus, G., Grabner, H., Reinfrank, F., Weninger, L., Pagano, D., & Maalej, W. (2013). An overview of recommender systems in requirements engineering. *Managing Requirements Knowledge*. Springer, 315-332.

[7] Felfernig, A., Jeran, M., Ninaus, G., Reinfrank, F., Reiterer, S., & Stettinger, M. (2014). Basic approaches in recommendation systems. *Recommendation Systems in Software Engineering*. Springer, 15-37.

[8] Fleckenstein, M., & Fellows, L. (2018). Data analytics. *Modern Data Strategy*. Springer, Cham, 133-142.

[9] Fu, W., Liu, J., & Lai, Y. (2018). Collaborative filtering recommendation algorithm towards intelligent community. *Discrete & Continuous Dynamical Systems-S*, 12(4 & 5), 811-822.

[10] Gautam, C.S., & Pandey, P. (2019). A review of big data environment, tools and challenges. *Journal of Emerging Technologies and Innovative Research*, 6(6), 569-575.

[11] Ghasemaghaei, M., Ebrahimi, S., & Hassanein, K. (2018). Data analytics competency for improving firm decision making performance. *The Journal of Strategic Information Systems*, 27(1), 101-113.

[12] Ghoneim, O.A. (2018). Traffic jams detection and congestion avoidance in smart city using parallel k-means clustering algorithm. In: *Proceeding of the International Conference on Cognition and Recognition.* Springer, 21-30.

[13] Hofmann, H.F., & Lehner, F. (2001). Requirements engineering as a success factor in software projects. *IEEE Software*, (4), 58-66.

[14] Horváth, I. (2007). Comparison of three methodological approaches of design research. In: *Proceeding of the 16th International Conference on Engineering Design*, 361-362.

[15] Horváth, I. (2019). Combining unmergeables: a methodological framework for axiomatic fusion of qualitative design theories. In: *Proceeding of the International Conference on Engineering Design*, Design Society, 3591-3600.

[16] Imran, M., Ahamad, M.V., Haque, M., & Shoaib, M. (2018). Big data analytics tools and platform in big data landscape. In: *Handbook of Research on Pattern Engineering System Development for Big Data Analytics,* IGI Global, 80-89.

[17] Jin, Z., Anderson, M.R., Cafarella, M., & Jagadish, H.V. (2017). Foofah: a programming-by-example system for synthesizing data transformation programs. In: *Proceeding of the 2017 ACM International Conference on Management of Data*, 1607-1610.

[18] Kalaian, S.A., Kasim, R.M., & Kasim, N.R. (2019). Descriptive and predictive analytical methods for big data. *Web Services: Concepts, Methodologies, Tools, and Applications*. IGI Global, 314-331.

[19] Li, Y. (2019). Utilizing dynamic context semantics in smart behavior of informing cyber-physical systems. *Doctoral dissertation*, Delft University of Technology.

[20] Lops, P., Jannach, D., Musto, C., Bogers, T., & Koolen, M. (2019). Trends in content-based recommendation. *User Modeling and User-Adapted Interaction*, 29(2), 239-249.

[21] Naaz, S., & Siddiqui, F. (2019). Application of big data in digital epidemiology. *Intelligent Systems for Healthcare Management and Delivery*. IGI Global, 285-305.

[22] Ricci, F., Rokach, L., & Shapira, B. (2011). Introduction to recommender systems handbook. In: *Recommender Systems Handbook*. Springer, 1-35.

[23] Saggi, M.K., & Jain, S. (2018). A survey towards an integration of big data analytics to big insights for value-creation. *Information Processing & Management*, 54(5), 758-790.

[24] Srinivasan, S., & Kumari, T.T. (2018). Big data analytics tools a review. *International Journal of Engineering & Technology*, 7(2.33), 685-687.

[25] Srivastava, S.K., & Roy, S.N. (2018). Recommendation system: A potential tool for achieving pervasive health care. *Next-Generation Mobile and Pervasive Healthcare Solutions.* IGI Global, 111-127.

[26] Wang, Y., Kung, L., & Byrd, T.A. (2018). Big data analytics: understanding its capabilities and potential benefits for healthcare organizations. *Technological Forecasting and Social Change*, 126, 3-13.