Journal of Intelligent Information Systems
https://doi.org/10.1007/s10844-018-0530-7

An overview of recommender systems in the internet of things

Alexander Felfernig[1] · Seda Polat-Erdeniz[1] · Christoph Uran[1] · Stefan Reiterer[1] · Muesluem Atas[1] · Thi Ngoc Trang Tran[1] · Paolo Azzoni[2] · Csaba Kiraly[3] · Koustabh Dolui[3]

Received: 20 March 2017 / Revised: 6 June 2018 / Accepted: 18 September 2018 /
Published online: 01 October 2018

Abstract
The Internet Of Things (IoT) is an emerging paradigm that envisions a networked infrastructure enabling different types of devices to be interconnected. It creates different kinds of artifacts (e.g., services and applications) in various application domains such as health monitoring, sports monitoring, animal monitoring, enhanced retail services, and smart homes. Recommendation technologies can help to more easily identify relevant artifacts and thus will become one of the key technologies in future IoT solutions. In this article, we provide an overview of existing applications of recommendation technologies in the IoT context and present new recommendation techniques on the basis of real-world IoT scenarios.

Keywords Recommender systems · Internet of things

1 Introduction

As an emerging paradigm, the Internet Of Things (IoT) (Atzori et al. 2010; Greengard 2015) represents a networked infrastructure connecting different types of devices in any place and anytime. IoT can be seen as the result of the convergence of the three main domains *Things*, *Internet*, and *Semantics*. Recommendation technologies (Jannach et al. 2010) can support the efficient identification of relevant artifacts and thus will become one of the key-technologies of IoT solutions. Recommender systems (Jannach et al. 2010) suggest items (alternatives, solutions) that are of potential interest for a user. Examples of related questions are: *which book should be purchased?*, *which test method should be applied*?, *which method calls are useful in a certain development context?* or *which apps (applications) are of potential interest for the current user?* A recommender system can be defined as *any system that guides a user in a personalized way to interesting or useful objects in a large space of possible options or that produces such objects as output* (Felfernig and Burke 2008).

✉ Seda Polat-Erdeniz
spolater@ist.tugraz.at

Extended author information available on the last page of the article.

Recommender technologies are mainly based on two fundamental approaches; collaborative filtering and content-based filtering. *Collaborative Filtering* (Konstan et al. 1997) is using the opinion of users with similar preferences whereas *Content-based Filtering* (Pazzani and Billsus 1997) is based on comparing the content of already consumed items with new items that can potentially be recommended to the user. Other basic recommendation approaches are *knowledge-based recommendation*, *group recommender systems*, and *hybrid recommendation*. *Knowledge-based recommender systems* (Felfernig et al. 2015) are based on explicit knowledge, rules or constraints about the item assortment, user preferences, and recommendation criteria (i.e., which item should be recommended in which context). *Group recommender systems* (Felfernig et al. 2018; Masthoff 2011) calculate recommendations where the whole group should be satisfied with the given recommendation. *Hybrid recommendation* (Burke 2002) combines basic recommendation approaches to compensate the weaknesses of individual ones. In Section 3, these basic approaches are explained in detail based on the real IoT use cases from our project.

In the IoT domain, recommendation functionalities are required, for example, in IoT workflow development, the recommendation of apps, and domain-specific scenarios such as food recommendation (Valtolina et al. 2014), personalized shopping (Magerkurth et al. 2011), and technology fairs (Munoz-Organero et al. 2010). Example upcoming IoT application domains are *health monitoring*, *animal monitoring*, *enhanced retail services*, *smart homes*, and *sports events* (Felfernig et al. 2016; Greengard 2015; Leitner et al. 2014; Ray 2015; Stolpe 2016).

As shown in Fig. 1, IoT sensors can be connected to an IoT gateway using various connection protocols such as 5G, BLE, LORA, and ZigBee. Users of the gateway can connect their gateway via WAN/LAN to manage/monitor their data and services. They can also manage/monitor the collected data by linking a cloud application with their IoT gateway.

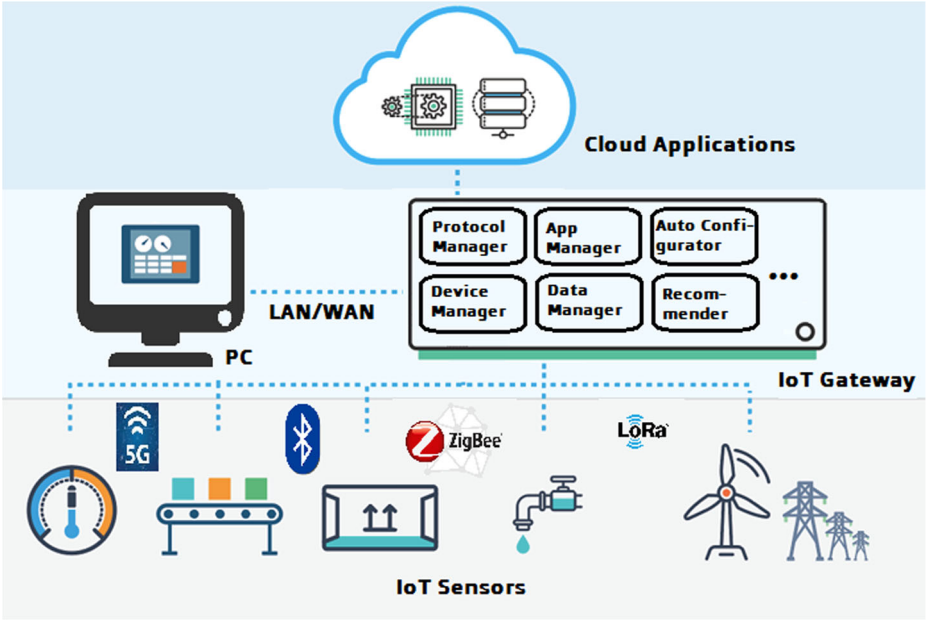

Fig. 1 Software architecture of an IoT gateway

An IoT gateway is a hardware and software-based solution, which, as its primary role, enables device-to-device and/or device-to-cloud communication. It is a platform to support connections between different data sources (sensors with various communication protocols) and destinations (local or remote data management entities, as well as various actuators). IoT gateways, positioned at the edge and near the devices, could also play a crucial role in the execution of services. A typical IoT gateway platform is composed of a device manager, a communication/data protocols manager, an application manager, and a data manager (see Fig. 1). Advanced IoT gateways contain additional functionalities among which a configurator and a recommender engine can be included to assist users in the configuration of the gateway or in recommending useful applications based on given gateway settings and user interaction protocols.

In health monitoring (so-called *Quantified-Self*) (Erdeniz et al. 2018; Maglogiannis et al. 2016; Menychtas et al. 2016), users need to know which measuring instruments are needed in their specific context and also how to change personal behaviors (e.g., eating and sports) to improve their situation. The realization of *Quantified-Self* concept requires the integration of several mobile health and IoT elements, where related applications are orchestrated around the IoT Gateway. The gateway connects to the home network and through the gateway's management user interface, the owner has access to all provided features, such as reporting and visualization tools, can manage (store/view/edit) their data and define an access policy to share data with their social network contacts. Wearable *activity trackers* and *medical sensors* (such as oximeters,[1] blood pressure monitors[2] or glucometers[3]) automatically communicate with the gateway whenever within range, and upload the most recent data. Integration with cloud platforms (such as Fitbit,[4] and GoogleFit[5]) allows data synchronization between the gateway and the owner's online profile, which enables the user to access their data through a web application. In addition, health and activity data can be downloaded to the gateway from the owner's personal accounts on relevant platforms.

In the context of *wildlife animal monitoring*, measuring devices and data collection units (typically drones) have to be selected and parametrized in such a way that the observation area is completely covered, i.e., the needed data can be provided in the required quality. IoT-based *retail services* are developed to support a personalized shopping experience in physical stores. In this context, recommender algorithms help to determine which offers should be recommended to a customer when, where, and in which format. In the context of *smart homes*, recommendation technologies improve the overall applicability of the installed equipment and can also help to optimize the usage of the available resources (e.g., minimizing power consumption). At *large scale sports events* such as marathons or triathlons, recommender systems can help the spectators to determine the current geographical location of certain athletes. This further results in recommended sites at which the athlete can be seen and cheered.

In the AGILE Project,[6] we have developed new recommendation approaches which are especially useful in IoT scenarios. The AGILE project aims to create Internet of Things

[1] https://en.wikipedia.org/wiki/Pulse_oximetry

[2] http://bestreviews.com/best-blood-pressure-monitors

[3] https://en.wikipedia.org/wiki/Glucose_meter

[4] https://www.fitbit.com/at/home

[5] https://www.google.com/fit/

[6] AGILE (An Adaptive & Modular Gateway for the IoT) is an EU-funded H2020 project 2016–2018 – see http://agile-iot.eu/.

(IoT) gateway technologies that support many devices, protocols, and corresponding management and development activities. The goal of this article is to show how recommenders can be applied in IoT scenarios and to propose new recommendation approaches for the IoT domain. In this context, we also provide an overview of existing applications of recommendation technologies in the IoT.

The remainder of this article is organized as follows. In Section 2, the state of the art of recommendation technologies in IoT application contexts is analyzed. In Section 3, we present a motivating example *"IoT for the smart home"* which is used throughout the paper for explaining the recommendation approaches. In Section 4, an overview of possibilities to apply basic recommendation approaches in the IoT domain is presented. In Section 5, we propose new recommendation approaches and explain them on the basis of example scenarios in the AGILE project. In Section 6, selection criteria for recommender algorithms are discussed. Finally, we discuss open research issues in Section 7 and conclude the article with Section 8.

2 Related work

Compared to other recommendation scenarios, IoT-based applications enable a deeper understanding of user preferences and behaviors which can primarily be explained by the availability of heterogeneous information sources (Amato et al. 2013). For instance, *personalized shopping* is a core element of IoT technology based retail environments (Magerkurth et al. 2011). Customers entering a store receive recommendations regarding items and corresponding price offers – these recommendations depend on the condition of the offered items. For example, if the expiration date of some of the offered items is approaching, and this information is detected via their RFID (radio frequency identification) tags (Finkenzeller 2010), corresponding special offers can be announced to the customer. Important IoT-related aspects are *automated quality control of items*, *context-dependent pricing*, and *targeted product information* (Mashal et al. 2016). The recommendation approach presented in Magerkurth et al. (2011) follows a knowledge-based (rule-based) paradigm. However, in this paper, we show how to generate such rules on the basis of sequence mining techniques (Srikant and Agrawal 1996).

Valtolina et al. (2014) introduce a *household scenario* where users ask for recommendations regarding recipes. In the context of an IoT infrastructure, a recommender system does not have to only rely on the preferences of the user but can take into account further information sources. For example, recipe recommendation can take into account the availability of food items in the fridge, personal diet plans, food consumption information from the last days, planned activities, and also historical data about last day's sports activities. In this scenario, the fridge can read the RFID tags (Finkenzeller 2010) of items and notify the mobile application over a BLE (bluetooth low energy) (Lee et al. 2007) connection. This availability of orthogonal data sources provided by IoT devices will help to increase the prediction quality of recommendation algorithms. In this scenario, the recommendations are based only on the data of the active user.

In many applications, such as recommending a vacation package, personalized content on a Web site, or a movie, it may not be sufficient to consider only users and items, it is also important to incorporate the *contextual information* into the recommendation process in order to recommend items to users under certain circumstances (Adomavicius and Tuzhilin 2015). Therefore, some recommender approaches focus on recommending the most relevant

items to users by taking into account any additional *contextual information*, such as time, location, or the company of other people (e.g., for watching movies or dining out). *Contextual information* is also important in many IoT use cases. Munoz-Organero et al. (2010) introduce *technology fairs* as a scenario where *context-aware recommender systems* can be applied. In such a scenario, users can receive information about exhibits of relevance and also be informed about lectures to attend depending on their personal preferences. In such scenarios, location tags (e.g. iBeacons (Martin et al. 2014)) are used to provide the location ID to the mobile applications over BLE.

Similar scenarios exist in domains such as museum visits of user groups, where recommendations can take into account aspects, such as time available, accessibility of objects at specific times, and personal interests. Visitors of museums are often overwhelmed by the information available in the space they are exploring. Therefore, finding relevant artworks to see in a limited amount of time is a crucial task. Such context-based recommender systems (Benouaret and Lenne 2015) for mobile devices adapt to the users profiles and is sensitive to their context (location, time, expertise, etc.). These recommenders improve the visitors' experience and help them build their tours on-site according to their preferences and constraints.

The authors of Cha et al. (2016), Martino and Rossi (2016), and Yavari et al. (2016) show how basic Artificial Intelligence (AI) approaches such as planning and clustering can be exploited for offering public services in a personalized fashion. They do not include basic recommendation approaches but, for example, scheduling and clustering that empower functionalities such as route planning and recommending points of interest.

Bahirat et al. (2018) present a data-driven approach to the development of a privacy-setting interface for IoT devices. By applying machine learning techniques to an existing dataset of users, a set of "smart" default profiles are generated. Using these smart profiles, privacy settings are recommended to users. The accuracy of their privacy-setting predictions is around 82%.

3 A motivating example: IoT for the smart home

The motivation for our study came from the needs of recommender systems for our IoT gateway project *AGILE*. We present a *IoT for the smart home* illustration in Fig. 2 as the motivating example throughout the paper.

In this smart home example, a user (*Alex*) installs an AGILE gateway to make his home smarter. After that, *Alex* buys a gas and a temperature sensor, connects them to his gateway then connects via the local network to the management user interface of his gateway on the web browser of his computer. At this stage, *Alex* needs to receive some *app*, *device* or *communication* protocol (BLE, zigbee, etc.) recommendations according to the overall settings on the gateway.

Moreover, when Alex wants to extend his smart home with a pollution monitoring system, he needs a special support for choosing the sensors and configuring the system properly. He may also need more help for the workflow development if he has very low knowhow about developing workflow with his new pollution monitoring sensors.

We explain how these recommendation needs for *apps*, *devices*, *protocols*, and *workflows/nodes* can be addressed based on Alex's smart home using recommendation technologies in Sections 4 and 5.

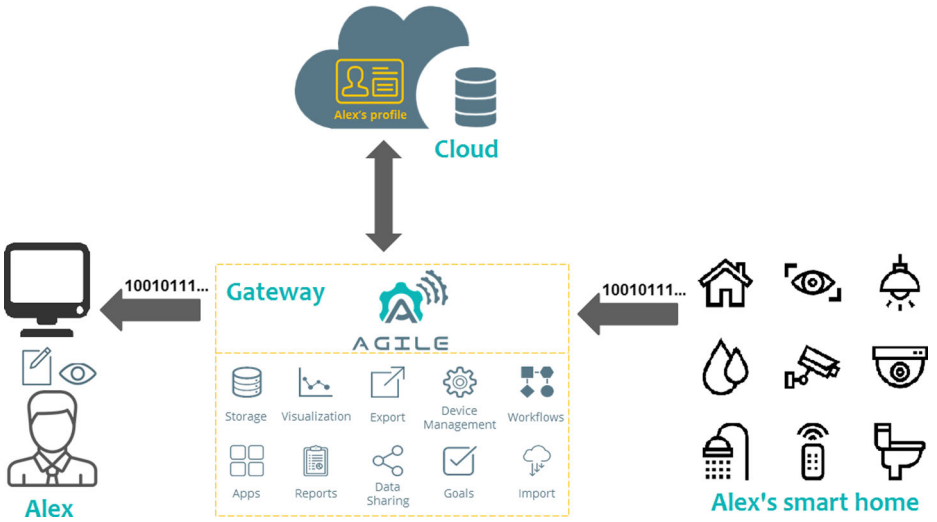

Fig. 2 An IoT based smart home on the basis of the AGILE gateway. All measured data from IoT devices (motion sensor, light sensor, etc.) are collected on the AGILE gateway over various connection protocols such as 5G, BLE, LORA, and ZigBee. Alex can monitor the collected data or edit gateway settings by connecting to his smart home gateway via WAN/LAN. He can install applications (e.g. fire alarm app) or connect various devices (e.g. gas sensor) to his AGILE gateway to extend his smart home. He can use a cloud service to export his smart home data to utilize powerful and online cloud based applications

4 Basic recommendation technologies in IoT

In the IoT context, recommender systems can support scenarios such as the recommendation of apps, services, sensor equipment, and IoT workflows (Felfernig et al. 2016). In the following subsections, we show how basic recommendation algorithms can be applied in IoT contexts. For a detailed discussion of underlying recommendation algorithms we refer to Jannach et al. (2010).

4.1 Collaborative filtering

Collaborative filtering (Konstan et al. 1997) is based on the idea of word-of-mouth promotion, i.e., the opinion of users with similar preferences plays a major role in a decision. These users are also denoted as *nearest neighbors*, i.e., users with similar preferences compared to the current user. The first step of a collaborative filtering recommender is to identify the *k-nearest neighbors*[7] and to extrapolate the preferences of the current user from the ratings of nearest neighbors using Formula (1).

$$similarity(item_a, item_b) = \frac{1}{1 + \Sigma_{i=1, i \in users}^{n} |eval(item_a) - eval(item_b)|} \quad (1)$$

In order to recommend new apps for *Alex's smart home gateway* (see Section 3), a recommender can provide a list of (additional) apps that can be of relevance on the basis of a given

[7] k represents the number of users with similar ratings compared to the current user.

Table 1 Collaborative filtering based app recommendation based on gateway profiles

Gateway profiles								
Devices					Apps			
Profile	Temp. sensor	Motion sensor	Gas sensor	Camera	Temp. alarm	Fire alarm	Thief alarm	Gas alarm
user1	1.0		1.0				1.0	
user2		1.0			1.0		1.0	
user3	1.0		1.0			1.0	1.0	
Alex	1.0		1.0		1.0			1.0

The apps *fire alarm* and *thief alarm* can be recommended to Alex since they are installed on gateways with the same devices available on the local gateway

local gateway configuration (e.g., devices and drivers). Collaborative filtering can determine such recommendations on the basis of gateway profile data collected in anonymous form. A simplified example of a related recommendation approach[8] is given in Table 1. In this context, information about configurations of other gateways is used to infer relevant apps to be additionally proposed/installed on the *local* gateway. In our example, the devices *temperature sensor* and *gas sensor* are connected to the *Alex's* gateway. The installation bases 1 and 3 (profiles) include all the devices also connected to the local gateway but include additional apps that are currently not installed on the local gateway (these are *fire alarm* and *thief alarm*). Consequently, these apps represent recommendation candidates for *Alex*.

There are a couple of similarity metrics used in the context of collaborative filtering scenarios for determining nearest neighbors (for details we refer to Jannach et al. (2010)). We also want to emphasize that the examples provided in this article are using basic recommendation approaches, for example, matrix factorization is a state-of-the-art algorithmic approach to support collaborative filtering. For the purposes of our examples, we introduce a simplified formula that supports the identification of *k-nearest neighbors*[9] (see Formula (1)). If applied to the example of Table 1, *profiles* are represented by *items* in Formula (1).

Alternatively, collaborative filtering can exploit the ratings ([0..5]) of users when interacting with an app marketplace (in this context, *users* have to be associated with the *items* contained in Formula (1)). The underlying idea is that gateways can be connected to app marketplaces where users can select and download apps that are of interest for their local installation. In this scenario, the evaluation data (ratings) of users serve as a basis for determining recommendations (see Table 2).

In Table 2, user 1 (the *nearest neighbor*) has provided app evaluations which are similar to those on *Alex's smart home gateway* (see Section 3). Consequently, a collaborative recommender proposes apps for *Alex* which have been investigated by the nearest neighbor but not by the current user (e.g., the *pollution monitoring* app).

[8] "1.0" denotes the fact that the device is installed on the corresponding gateway (profile).

[9] For simplicity we assume $k = 1$.

Table 2 Collaborative filtering based app recommendation based on user ratings

	User ratings to apps							
	Temp. alarm	Fire alarm	Thief alarm	Gas alarm	Heart activity	Running app	Poll. mon.	Dog mon.
user1	1.0		4.5				3.0	
user2		2.0			4.5		2.5	3.5
user3	4.0		3.0			2.5	3.0	
Alex	1.0		4.0	4.0				

user-1 is the nearest neighbor of Alex. The *pollution monitoring* app has been rated by *user-1* but has not been rated by Alex and is therefore recommended

4.2 Content-based filtering

Content-based filtering (Pazzani and Billsus 1997) compares the content of already consumed items with new items that can potentially be recommended to the user, i.e., to find items that are similar to those already rated positively by the user using Formula (2).

$$similarity(user, item) = \frac{features(user) \cap features(item)}{features(user) \cup features(item)} \tag{2}$$

Another alternative for app recommendation is to implement a content-based recommendation approach where apps can be recommended for installation if their required devices (it is assumed that this information is given for each app) are "compatible" with the local gateway configuration (profile).[10]

When applying a content-based filtering based approach, recommended items are determined on the basis of the similarity between the local gateway profile information (e.g., in terms of installed devices) and the profile information of apps available, for example, on a marketplace in the cloud. Similar to collaborative filtering, there are different types of similarity metrics (see, e.g., Jannach et al. 2010). For the purposes of our examples, we introduce a simplified formula that supports the identification of, for example, relevant apps for the local gateway.

When we apply Formula (2), *user* is Alex's gateway, *item* is each app, and features are the devices of each app and the gateway. Therefore, it determines the similarity on the basis of the information about installed devices.[11] However, this approach can be extended to include further information, for example, regarding installed modules and network protocols available on the gateway. In our example of Table 3, the *fire alarm* app has the highest similarity with the profile information of *Alex's smart home gateway* (see Section 3), therefore this app is recommended.

Note that content-based recommendation is often applied when the similarity between textual information from different sources has to be determined. Therefore, the approach presented in this article can be extended to the matching of text-based search criteria and the textual description of apps.

[10]Please also note that it is possible to support scenarios where apps are recommended that do require additional hardware/device driver components in order to work properly.

[11]"1.0" denotes the fact that the device is installed on the corresponding gateway (profile).

Table 3 Content-based app recommendation

Apps	Devices					
	Temp. sensor	Camera	Gas sensor	ZigBee	BLE	WiFi
Thief alarm		1.0			1.0	1.0
Fire alarm	1.0		1.0		1.0	1.0
Temp. alarm	1.0			1.0		
Alex's gateway	1.0		1.0		1.0	

According to Formula (2), the *fire alarm app* has the highest similarity to the local gateway profile

4.3 Utility-based recommendation

Utility-based recommendation (Felfernig and Burke 2008) is based on the idea that – given a set of items – item ranking is determined on the basis of multi-attribute utility theory (MAUT) (Winterfeldt and Edwards 1986). In this case, each item is evaluated with regard to a set of interest dimensions. In the context of optimizing the used data transfer protocols, example dimensions could be *efficiency* (measured in terms of transfer rates) and *economy* (measured in terms of costs for data connections). Utility-based recommendation is often combined with knowledge-based recommendation. In this context, customer-individual preferences can also be learned by analyzing existing user interaction data (Jannach et al. 2010).

Modern embedded systems included in IoT scenarios support a rich set of connectivity solutions (e.g., 3G, LTE, TD-LTE, FDD-LTD, WIMAX, and Lora). In this context, recommendation technologies play an important role when it comes to suggesting the best connectivity configurations for the selected communication channel. The recommendation can be based, for example, on location information, available connectivity, performance and reliability requirements, and contractual aspects and costs. Gateway configurations can be manually defined by users but can also be determined on the basis of a configurator that is in charge of keeping the overall system installations consistent. A configurator (e.g., a constraint solver) can determine alternative configurations which have to be ranked. In order to determine a ranking for alternative configurations, a MAUT-based approach can be used. Examples of evaluation *dimensions* (dim) used in MAUT could be *performance*, *reliability*, and *costs* using Formula (3). Depending on the current gateway configuration and the usage context, a configurator can determine alternative re-configurations and rank them accordingly.

$$utility(item, user) = \Sigma_{d \in dim} interest(user, d) \times value(item, d) \tag{3}$$

An example of the application of a utility-based approach is the following. Table 4 includes an example evaluation of connectivity protocols (*BLE* and *ZigBee*) to be used on the gateway. Furthermore, Table 5 includes the personal preferences of *Alex*.

Table 4 Utilities of protocols

Protocol	Performance	Reliability	Costs
BLE	9	5	2
ZigBee	5	8	3

Table 5 User preferences w.r.t. *performance*, *reliability*, and *costs* (in between [0..1])

User	Performance	Reliability	Costs
Alex	1.0	0.3	0.1

In order to recommend a connectivity protocol for *Alex's smart home gateway* (see Section 3), we can apply a utility function (see, e.g., Formula (3)). When we apply the utility function, *item* stands for a protocol, and dimensions (*dim*) are the protocol utilities. Therefore, *BLE* (utility(*BLE*,Alex)=10.7) is recommended rather than *ZigBee* (utility(*ZigBee*,Alex)= 7.7) to Alex because the utility value of *BLE* for Alex is higher than the utility value of *ZigBee* for Alex.

4.4 Group recommender systems

Group recommender systems (Felfernig et al. 2018) are based on the idea that recommendations are not determined for a single user but the whole group should be satisfied with the given recommendation (e.g., a family's decision regarding a smart home solution). Recommendations in this context are often determined on the basis of group decision heuristics. For example, *least misery* is a heuristic that prefers recommendations with the property that the misery of all group members is minimized as shown in Formula (4). In contrast, *most pleasure* tries to maximize the pleasure of individual group members. Also in the context of group recommender systems, hybrid approaches can be developed, i.e., individual group recommendation heuristics can be combined with each other.

$$LM = \max_{(t \in I)}(min(t)) \tag{4}$$

In scenarios where a group of users is in charge of making a decision, group recommenders can provide support (Masthoff 2011). For example, if Alex shares his smart home with two home mates *Bob* and *Tom*, this group of smart home users is in charge of selecting an appropriate smart home solution for the smart home where they live together. User-specific evaluations of different smart home theft protection solutions on Google Playstore[12] are depicted in Table 6.

Let's apply the group recommendation algorithm *least misery* (see Formula (4)) on given group ratings. In Formula (4), t is an *item*, I a set of *items*, and LM is assumed to return a recommended item for the group. If we apply *least misery* group recommendation, *SalientEye Home Security Alarm* is recommended to the group since the minimum rating for this item is 4.0 (which globally represents the best least misery value for all group members).

4.5 Hybrid recommendation

Hybrid recommendation (Burke 2002) is based on the idea of combining basic recommendation approaches in such a way that one helps to compensate the weaknesses of the other.

For example, when combining content-based filtering with collaborative recommendation, content-based recommendation helps to recommend unrated items. If a user has already consumed some items (e.g., purchased some IoT apps), the content description of a new item can be compared with the descriptions of items already purchased by the user. If

[12] https://play.google.com/store/apps

Table 6 Selecting a smart home theft protection solution for Alex and his home mates (Bob and Tom)

	Group ratings to smart home apps			
	SalientEye home security alarm	Yale smart living home	AtHome camera home security video surveillance	Home security camera - Alfred
Bob	5.0	4.0	5.0	3.0
Tom	4.0	5.0	3.0	3.0
Alex	4.0	3.0	4.0	3.0

SalientEye Home Security Alarm is the best option for the group according to the *least misery* recommendation

the new item is similar to some of the already consumed ones (e.g., installed apps), it can be recommended to the user. Combining the recommendations of different algorithms, for example, on the basis of a voting mechanism, can help to significantly increase prediction quality (Jannach et al. 2010).

In the AGILE project, we have developed a hybrid recommender. A workflow recommendation is calculated based on the contents of the active workflow, devices connected to the user's gateway, and other similar gateway profiles (their nodes, workflows, and devices). We have combined the recommendation results of aforementioned basic approaches content-based filtering, and collaborative filtering with the recommendation results of our new approach SeqReq: Sequences based Recommendation (see Section 5.1). As shown in Fig. 3, the AGILE NodeRed[13] development environment presents recommendation results of workflows and nodes.

Node-Red[14] workflows are composed of connected nodes where the order and type of the nodes are important. One node sends its output to the next connected node as an input. For example, in Fig. 4,[15] the *email* node can not be used before the *openweathermap* node in the workflow because it needs an input (*the email text*) coming from the connected predecessor *openweathermap* and *function* nodes.

5 Advanced recommendation approaches in IoT

In this section, we introduce three new recommendation approaches which go beyond the basic ones introduced in Section 3. These approaches have been developed to support specific requirements of the use cases in the AGILE project.

5.1 SeqReq: Sequences based recommendation

In the AGILE project, in situations where sequences of recommendable items play important roles, we required an alternative recommendation approach compared to the basic ones introduced in Section 3. Thus, we have developed SeqReq (Sequences based Recommendation) which recommends items based on sequential pattern mining. In this subsection, we first explain sequential pattern mining, then show on the basis of a node recommendation

[13] https://nodered.org/
[14] https://nodered.org/
[15] http://developers.sensetecnic.com/article/a-node-red-flow-to-monitor-the-weather/

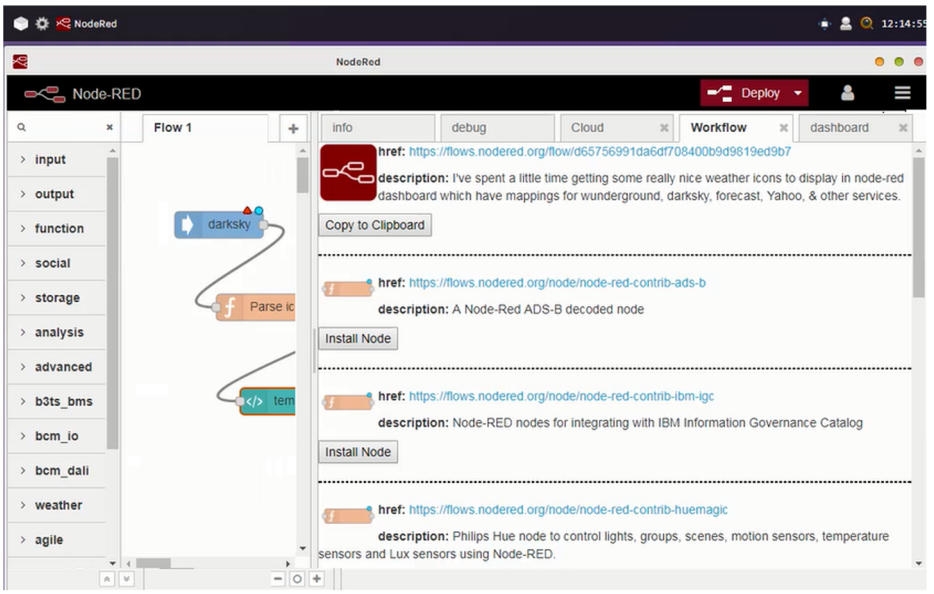

Fig. 3 The AGILE Node-Red development environment of *Alex's smart home gateway* (see Section 3). Hybrid recommendation is applied to recommend workflows and nodes which are the displayed in the *workflow tab*

example from the AGILE Node-Red[16] development environment how SEQREQ can be used based (see Fig. 3).

Sequential pattern mining (Han et al. 2001; Zaki 2001) is used for finding sequential patterns (sequences). Various algorithms have been proposed to find such patterns. For example, when analyzing user behaviors, a sequential pattern mining algorithm may find that many customers also buy a gas sensor, after having purchased a temperature sensor. Therefore, buying a gas sensor after a temperature sensor is a common sequence as: *[temperature sensor, gas sensor]*.

SEQREQ is useful in cases where the sequence of items are important. We explain SEQREQ on the basis of a workflow development use case from the AGILE project where sequences (or orders) are very important. In such a setup, we can find many common node sequences in the workflow repository. By searching common sequences of nodes in a workflow repository, we can build a look-up table with sequences an their occurrence frequencies (the number of observations) and distances (the number of links between two nodes). For example, in Fig. 4, the distance (the number of links) between the nodes *openweathermap* and *email* is 2. If there is another workflow in the repository which includes the nodes *openweathermap* and *email* with a distance 4, then the average distance of the nodes become $(2 + 4) \div 2 = 3$. Table 7 includes an example of the application of pattern mining in the context of workflows and nodes. This example is using only sequences with two nodes and an active workflow which has any number of (> 1) nodes.

The recommender's output is a list of nodes according to Formula (5). In this formula, $isFirstNodeInAW_{sequence-x}$ is 1 if the first node of sequence-x is observed in the active workflow (AW), otherwise it is 0. $Freq_{sequence-x}$ is the frequency of sequence-x

[16] http://agile-iot.eu/wiki/index.php?title=How_to_develop_an_App

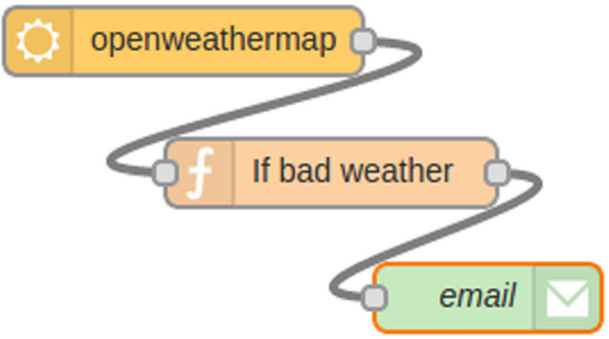

Fig. 4 An example sequence of nodes in a workflow. The first node *openweathermap* collects the weather forecast from https://openweathermap.org/ and provides the data to the next the node. The next node *function* (which is named as "*if bad weather*") checks the forecast data whether it is clear or not (rainy, snowy, or stormy). If the forecast says the weather is not clear, this *function* outputs a message to the next connected node. Finally, the next node (*email*) sends the received input message by email to a specified email address

and $Dist_{sequence-x}$ is the distance between the nodes in sequence-x. When this formula is applied on Table 7, the list of the recommended nodes for Alex is {*MSSQL, tingodb, ovh_sms*}. These recommended nodes are from the sequences with $Similarity > 0$.

$$Similarity(\text{sequence-x}, AW) = \frac{Freq_{\text{sequence-x}}}{Dist_{\text{sequence-x}}} \times isFirstNodeInAW_{\text{sequence-x}} \quad (5)$$

5.2 ConfReq: Recommendations for configurators

Existing recommendation technologies focus on simple items, however there is also a need for recommendation technologies for complex items. In the AGILE project, we have developed recommendation technologies to support configuration process for complex products and services. ConfReq provides recommendations of search heuristics to improve runtime performance of configurators. In this subsection, first we explain the relationship between configuration technologies and constraint satisfaction problems, then show how ConfReq can be applied on a constraint satisfaction problem from the smart home domain.

The configuration of a new gateway infrastructure requires *configuration technologies* with integrated recommendation functionalities (Falkner et al. 2011). When starting a new

Table 7 Sequential patterns (sequences) of nodes from a Node-Red repository

	Nodes	Frequency	Distance	Similarity
sequence-1	[mqtt in, MSSQL]	112	5	22.4
sequence-2	[mqtt in, tingodb]	78	12	6.5
sequence-3	[http out, twitter]	102	8	0
sequence-4	[e-mail, ovh_sms]	27	9	3
sequence-5	[openweathermap, email]	72	3	0
AW	[mqtt in, http in, email]	–	–	

Frequency is the number of observations of a sequence in the repository and *Distance* is the average distance (number of links) between the nodes in related workflows. AW is the active workflow on *Alex's smart home gateway* (see Section 3)

configuration, the configuration environment should be able to exploit information about already existing gateway installations in order to reuse some parts of the installation for the new gateway. In this context, recommendation technologies have to be integrated in order to guide search. Similar requirements exist in reconfiguration scenarios where the system has to react on changes in the set of installed applications or connected sensors. In such situations, for example, data transfer protocols have to be adapted in order to optimally take into account changes in the operating environment.

Constraint Satisfaction Problem. A configuration problem can be formulated as a *constraint satisfaction problem* (*CSP*) (Tsang 1993) which is defined as a triple *(V,D,C)* where V is a set of variables, D is set of domains for each variable, and C is a set of constraints. The constraint set also includes the user requirements (REQ) if already provided by the user. In order to find a solution for a CSP, there should be a correctly formed configuration task (see Definition 1).

Definition 1 (Configuration Task.) A configuration task can be defined as a $CSP(V, D, C)$. $V = \{v_1, v_2, .. , v_n\}$ represents a set of finite domain variables. $D = \{\text{dom}(v_1), \text{dom}(v_2), .. , \text{dom}(v_n)\}$ represents a set of variable domains, where $\text{dom}(v_k)$ represents the domain of variable v_k. $C = C_{KB} \cup REQ$ where $C_{KB} = \{c_1, c_2, .. , c_q\}$ is a set of domain specific constraints (the configuration knowledge base) that restricts the possible combinations of values assigned to the variables in V. $REQ = \{c_{q+1}, c_{q+2}, .. , c_t\}$ is a set of customer requirements, which is also represented as constraints.

ConfReq supports CSP solvers by recommending heuristics based on *cluster specific heuristic* (Erdeniz et al. 2017). ConfReq can support the configurator by recommending *where to start the solution search*. In order to improve the runtime performance of CSP solvers, search is guided by so-called variable and value ordering heuristics. ConfReq has been introduced to increase the runtime performance of CSP solvers based on learned variable ordering heuristics. ConfReq clusters past $REQs$ using k-means clustering (see Formula (6)) and calculates variable ordering heuristics for each cluster using a genetic algorithm. Therefore, the heuristic of a cluster can be used for a new CSP (which is closer to this cluster rather than other clusters). It has been shown that ConfReq significantly improves the runtime performance of the used CSP solver (Choco Solver (Prud'homme et al. 2016)).

For example, when *Alex* (see Section 3) wants to extend his smart home's pool with new IoT based sensors or a IoT based pump, he needs a professional support for installing the suitable devices and configuring them. Therefore, we developed a configuration recommender using ConfReq to increase the runtime performance of CSP solvers by the help of *the recommendation of heuristics*.

His swimming pool can be configured with one of the available pool pumps as given in Table 8. ConfReq first clusters past user requirements based on K-means clustering.

Table 8 Product table with five types of pool pumps where each has three features (power in Watt, price in Euro, and size in Centimeter)

	pump1	pump2	pump3	pump4	pump5
v1 (power)	1000	1000	600	600	1200
v2 (price)	120	140	100	100	160
v3 (size)	12	8.7	14.5	7.2	11

Table 9 Six sets of user requirements (*REQs*) stored in previous configuration sessions

REQ1	REQ2	REQ3	REQ4	REQ5	REQ6
v1=1200	v1=1000	v1=1200	v1= –	v1= –	v1= –
v2=160	v2=140	v2= –	v2=100	v2= –	v2= 160
v3= –	v3= –	v3= –	v3= –	v3= 7.2	v3= –

Variables, shown with "–", are not assigned by the user

K-means clustering is based on the minimization of distances between the cluster elements and the mean values of clusters as shown in Formula (6), in this context k is the number of target clusters, S is a cluster set, μ_i is the average value of cluster elements in the S_i and x is a cluster element in S_i.

$$min \sum_{i=1}^{k} \sum_{x \in S_i} \| Distance(x, \mu_i) \|^2 \tag{6}$$

Euclidean n-distance is generally applied by K-means clustering as the distance measurement equation between the cluster elements and mean values of clusters as shown in Formula (7) where x_i is the i^{th} attribute in the cluster element x, y_i is the i^{th} attribute in the cluster element y, and n is the number of attributes in one cluster element.

$$Distance(x, y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2} \tag{7}$$

In our example, we use k-means clustering with *number of clusters: k=2* as shown in Table 9. Then, we obtain the clusters as shown in Table 10. ConfReq applies *Min-Max Normalization* (see Formula (8)) on REQs before clustering.

$$v_{i\_norm} = \frac{v_i - dom(v_i)_{min}}{dom(v_i)_{max} - dom(v_i)_{min}} \tag{8}$$

After calculating the clusters, ConfReq learns cluster specific variable ordering heuristics using supervised learning (Venturini 1993) based on a genetic algorithm (GA) (Erdeniz et al. 2017) which uses a fitness function (see Formula (9)). This formula minimizes the total runtime (τ) for finding the first solution for a CSP over all sets of user requirements in a cluster.

$$min \left(\tau = \sum_{i=1}^{n} runtime(solve(CSP_i)) \right) \tag{9}$$

An *individual* is generated by the genetic algorithm in the form of an array which includes all variables of the CSP. According to the defined parameters of the genetic algorithm (the most important ones are: the maximum number of generations, and the mutation

Table 10 Two clusters generated for six sets of user requirements (see Table 9)

cluster1		cluster2	
Items	Centroid	Items	Centroid
REQ1,REQ2,REQ3	[1133,100,0]	REQ5,REQ6,REQ4	[0,86.66,2.4]

rate), it finds an individual which is the best of all individuals in the generated population (see the fitness function Formula (9)).

Now, let us assume that the cluster-specific variable ordering heuristics are learned as follows: *h1:[v1,v2,v3]* for cluster1 and *h2:[v1,v3,v2]* for cluster2. *REQ_new*={*v1=1000*} is a new set of user requirements based on the same pool-configuration problem and it is the closest to cluster1 according to Formula (7). Consequently, we use the variable ordering heuristic *h1:[v1,v2,v3]* for solving the corresponding CSP. Using this heuristic, the CSP solver searches for a solution by instantiating the variables in the order of *h1*, which results in the product *pump1:(v1=1000,v2=1200,v3=1200)*. With the help of learned heuristics, a solution can be found in a shorter time. A more detailed discussions of the experimental results of our approach is presented in Erdeniz et al. (2017).

5.3 DIAGREQ: recommending diagnoses

In the AGILE project, we also developed recommender algorithms that provide support in *inconsistent situations*. For this purpose, we developed DIAGREQ that provides recommendations for resolving inconsistencies effectively in terms of *runtime* and *accuracy*. In this subsection, first we introduce diagnosis problems, then show how DIAGREQ can be applied on the basis of an example diagnosis problem about pollution-monitoring systems.

In the context of knowledge-based configuration, search interfaces allow the specification of requirements (REQ) and display a solution if the requirements are consistent with the recommendation knowledge base. However, in many cases no solution exists for a given set of requirements and the user needs support in finding a way out of the *"no solution could be found dilemma"* (Jannach et al. 2010).

In this context, diagnoses can be recommended that help to solve inconsistencies. When a constraint set is inconsistent, there is no solution for the corresponding constraint satisfaction problem (CSP) / configuration problem. In this case, a diagnosis algorithm can be applied to find the cause of the inconsistency. A corresponding *Customer Requirements Diagnosis Problem (REQ Diagnosis Problem)* and *Customer Requirements Diagnosis (REQ Diagnosis)* can be defined as follows:

Definition 2 (REQ Diagnosis Problem.) A customer requirements diagnosis problem (REQ Diagnosis Problem) is defined as a tuple (C_{KB}, REQ) where $REQ = \{c_1, c_2, ..., c_m\}$ is the set of given customer requirements and C_{KB} represents the constraints part of the configuration knowledge base.

Definition 3 (REQ Diagnosis.) A REQ diagnosis for a REQ diagnosis problem (C_{KB}, REQ) is a set $\Delta \subseteq REQ$, s.t. $C_{KB} \cup (REQ - \Delta)$ is consistent. $\Delta = \{c_1, c_2, .., c_n\}$ is minimal if there does not exist a diagnosis $\Delta' \subset \Delta$, s.t. $C_{KB} \cup (REQ - \Delta')$ is consistent.

For example, when *Alex* (see Section 3) wants to extend his smart home gateway with pollution monitoring sensors, he needs a professional support for installing and configuring these sensors to his home. Therefore, we developed a ramp-up configurator where Alex is responsible for defining the requirements and the ramp-up configurator finds a solution according to Alex's requirements. The ramp-up configurator uses DIAGREQ to solve inconsistency situations in a reasonable time with a high prediction quality. In this example, the knowledge base is a product catalog (see Table 11).

Moreover, we know about six previous monitoring station requirement specifications that lead to an inconsistency, i.e., no solution could be found (see Table 12). Furthermore, this

Table 11 Five types of air pollutant sensors where each has three features as sensitivity (ppm: parts per million), price (euro), and size (millimeter)

	SO_2 sensor	NO sensor	PH_3 sensor	CO sensor	H_2S sensor
v1 (sensitivity)	1000	1000	600	600	1200
v2 (price)	14	12	10	10	16
v3 (size)	12	8.7	14.5	7.2	11

Sensors in the product table measure the level of the following pollutants; SO_2: Sulfur Dioxide, NO: Nitric Oxide, PH_3: Phosphine, CO: Carbon Monoxide, H_2S: Hydrogen Sulfide

table includes the information of selected products by other users after changing their initial inconsistent requirements (see Table 13).

In order to find a diagnosis recommendation, we developed DIAGREQ (Atas et al. 2017) which can find diagnoses using cluster-specific constraint ordering heuristics that are used by direct diagnosis search. In order to find heuristics, at first, DIAGREQ clusters the inconsistent requirements (see Table 12) using k-means clustering. It applies *Min-Max Normalization* (Visalakshi and Thangavel 2009) on REQs (see Formula (8)). After clustering the normalized REQs, DIAGREQ uses supervised learning based on a genetic algorithm) to find constraint orderings. An individual generated in the genetic algorithm is represented as array of constraints of the CSP (e.g. *[c2,c3,c1]*). According to the defined parameters of the genetic algorithm (maximum number of generations, mutation rate, etc.), it finds an individual which is the best one among the other individuals in the generated population. This individual has the best fitness value which is calculated using the fitness function (see Formula (10) and Formula (11)).

$$min\left(\tau = \sum_{i=1}^{n} runtime(\Delta_i)\right) \tag{10}$$

$$max\left(\pi = \frac{\text{k=\#(correct predictions)}}{n = \text{\#(predictions)}}\right) \tag{11}$$

In supervised learning, runtime and accuracy of a diagnosis are calculated using the requirement specifications in Table 12 and selected solutions (see Table 13). Based on supervised learning, the corresponding learned constraint orderings are shown in Table 14.

Whenever Alex generates a new inconsistent requirements set (see Table 15), DIAGREQ can be applied to restore the consistency. First, it finds the closest cluster to this new requirement set which is *cluster1* and applies this cluster's constraint ordering before running the diagnosis algorithm. For example, for a accuracy-efficient solution we apply $H_1\pi$ shown in Table 15. Then, DIAGREQ finds the diagnosis Δ. After eliminating Δ from REQ_new, *REQ_new_diagnosed* is obtained as shown in Table 15. Now a CSP solver can solve the

Table 12 Inconsistent requirements (REQs) collected from previous configuration sessions of different users

	user1 REQ1	user2 REQ2	user3 REQ3	user4 REQ4	user5 REQ5	user6 REQ6
c1	v1=1200	v1=1000	v1=1200	v1=600	v1=600	v1=1000
c2	v2=10	v2=16	v2=14	v2=12	v2=0	v2=12
c3	v3=12	v3=14.5	v3=11	v3=12	v3=11	v3=7.2

Table 13 Sensors finally selected by users who install the monitoring station (after their specified inconsistent requirements given in Table 12)

user1	user2	user3	user4	user5	user6
SO_2 sensor	SO_2 sensor	NO sensor	H_2S sensor	CO sensor	NO sensor

CSP of *REQ_new_diagnosed*. Finally, a CSP solver can find two solutions from the product catalog (see Table 11): *PH_3 sensor* and *CO sensor*. Since we used the accuracy-based heuristics to diagnose this problem, the solutions have high probabilities to be accepted by Alex. Related experimental results are presented in our previous work (Atas et al. 2017).

6 Selection of recommendation algorithms

The five basic approaches of collaborative filtering (CF), content-based filtering (CBF), knowledge-based recommendation (KBR) (utility-based recommendation is also considered as a subtype of knowledge based recommendation since the utility function is indeed a utility constraint (Felfernig et al. 2010)), and group recommendation (GR) are based on different knowledge sources and also have different strengths and weaknesses – a corresponding overview is shown in Table 16.

Easy setup Collaborative filtering and content-based filtering systems are easy to set up since only basic information about item names, descriptions, and graphical representations is needed – the same holds for group recommender systems which rely on pre-defined heuristics to determine recommendations. Knowledge-based recommender systems require a more detailed specification of the recommendation knowledge (represented in terms of attributes, constraints, and/or similarity metrics) and also of the corresponding items (semantic properties have to be specified).

Conversational approach Both, group recommender systems and knowledge-based recommender systems are often based on a conversational approach where users have to provide answers to questions (preferences regarding the properties of alternatives) and recommender systems propose solutions (candidate items) which serve as a basis for further user feedback. Critiquing-based recommender systems (Burke et al. 1997) support the specification of critiques which represent user feedback on the properties of an item currently shown to the user. Constraint-based recommender systems allow the specification and re-specification of preferences (similar to the concept of critiques) and then support users in

Table 14 Learned constraint ordering heuristics (H)

κ_1	$H_1\pi$:	{c3, c2, c1}
	$H_1\tau$:	{c2, c3, c1}
κ_2	$H_2\pi$:	{c1, c3, c2}
	$H_2\tau$:	{c1, c2, c3}

Runtime (τ) and precision (π) are calculated for each cluster (κ_i)

Table 15 Alex generates a new inconsistent set of requirements (REQ_new) which needs to be diagnosed

	REQ_new	REQ_new_reordered	Δ	REQ_new_diagnosed
c1	v1=1200	v3=10	v3=10	–
c2	v2=10	v2=10	–	v2=10
c3	v3=10	v1=1200	v1=1200	–

Its constraints are reordered using $H_1\pi$ before diagnosis. Then, $REQ\_new\_diagnosed$ is obtained by eliminating Δ. When the corresponding CSP of $REQ\_new\_diagnosed$ is solved, the solutions PH_3 *sensor* and *CO sensor* are found

situations where no solution can be identified (Felfernig and Burke 2008). Collaborative filtering and content-based recommendation approaches are typically not used in the context of conversational scenarios.

Adaptivity Both, collaborative filtering and content-based recommendation are more adaptive in the sense that new ratings provided by users are automatically taken into account. Knowledge-based recommendation does not support this type of adaptivity since utility schemes (Winterfeldt and Edwards 1986) are in most of the cases adapted manually, i.e., are not learned. Group recommender systems in their basic form (Felfernig et al. 2018) do not take new evaluations of items into account.

Serendipity effects Serendipity characterizes a situation where a user is confronted with relevant items he/she did not expect. Serendipity effects can be achieved primarily using collaborative filtering and variants thereof (Koren et al. 2009). Since content-based recommendation in its basic from does not take into account the preferences of other users, less serendipity effects can be achieved with this approach. Serendipity effects can be somehow achieved with knowledge-based recommenders, however, in this context serendipity knowledge has to be encoded into the underlying recommendation knowledge base. In critiquing-based systems, this encoding is part of the similarity metrics used to determine new candidate items. Also in basic types of group recommender systems, the serendipity rather depends on the encoding in corresponding group decision heuristics (Masthoff 2011).

Ramp-up problems Ramp-up problems occur if a recommendation algorithm relies on initial information which is sometimes not available. For example, in collaborative filtering, user preferences have to be available in terms of item ratings – if these ratings are

Table 16 Selection criteria for recommendation algorithms

Algorithms				
Algorithm	CF	CBF	KBR	GR
Easy setup	Yes	Yes	No	Yes
Conversational	No	No	Yes	Yes
Adaptivity	Yes	Yes	No	No
Serendipity effects	Yes	No	Yes	No
Ramp-up problem	Yes	Yes	No	No
Transparency	No	No	Yes	No
High involvement items	No	No	Yes	Yes

not available, no recommendations can be determined. Ramp-up problems primarily exist in the context of collaborative filtering and content-based recommendation. In collaborative filtering, users have to rate items in order to enable the algorithm to determine nearest neighbors. In content-based recommendation, users have to specify which kinds of items are perceived as interesting in order to enable the algorithm identify items with similar characteristics. Knowledge-based recommendation does not have to deal with ramp-up problems since the recommendation knowledge is already pre-specified (in terms of constraints, rules, or similarity metrics). Similarly, group recommenders do not have a ramp-up issue since recommendation calculation is based on pre-defined decision heuristics.

Transparency Transparency is a measure that specifies to which extent recommendations can be explained to users. In collaborative filtering, explanations are based on the similarity to nearest neighbors (*this item is recommended since similar users also purchased this one*). Explanations in content-based recommendation scenarios are based on the similarity between the recommended item and those already consumed by the user (*this item is recommended since you purchased similar items in the past*). In both cases, explanations can be regarded as shallow, i.e., do not provide deep insights to the reasons of a specific item recommendation. Group-based recommender systems generate explanations that strongly depend on the used heuristics, for example, *this item is recommended to the group since no misery can be expected by one of the group members* (Felfernig et al. 2017). The highest degree of transparency can be expected from knowledge-based recommendation approaches where solutions can be explained on the basis of information gained from the underlying reasoning process. Especially in the context of constraint-based recommendation, it is possible to generate explanations that help to understand as to why no solution could be identified (Felfernig and Burke 2008).

High involvement items High-involvement items (Felfernig et al. 2017) are items that are selected and/or purchased in most of the cases after a careful consideration since the impact of suboptimal decisions can be rather high. Examples of related items are IoT based smart homes, IoT based animal monitoring stations, and IoT based pollution monitoring stations. Both, collaborative filtering and content-based filtering are used in most of the cases for recommending low-involvement items such as IoT apps, and IoT sensors. Group recommender systems are exploited for scenarios ranging from decisions such as choosing a smart home alarm system to complex products such as configuring a new pollution monitoring station. Ratings related to high-involvement items are provided less frequently which makes collaborative filtering and content-based recommendation less applicable (for example, preferences regarding an apartment or a car could significantly change over time).

Recommender libraries Recommendation algorithms and heuristics are in many cases regarded as a central intellectual property of a company and are therefore often not implemented on the basis of existing recommendation libraries. *Strands*[17] is a commercial recommendation library supporting different types of recommendation algorithms for the retail and finance sector. *MyMediaLite*[18] is a .NET based recommendation library that supports collaborative filtering. *LensKit*[19] is a toolkit from the University of Minnesota

[17] strands.com.
[18] mymedialite.net.
[19] lenskit.org.

that supports different kinds of collaborative filtering algorithms. *Movielens*[20] is a related non-commercial movie recommendation platform – it also provides a couple of publicly available datasets that can be exploited for the evaluation of the predictive quality of recommendation algorithms. *Apache Mahout*[21] is a machine learning environment that also includes different types of collaborative filtering approaches. *Choco*[22] is an example of an open-source constraint library that can be exploited, for example, for the development of constraint-based recommender applications. Another example of a constraint-based recommendation environment is *WeeVis*[23] that supports the integration of constraint-based recommender applications into Wiki pages. Finally, *Choicla*[24] is a group recommender environment that supports group decision making for non-configurable items.

7 Research issues

Scalability of algorithms In some scenarios, recommendation algorithms can be deployed in the cloud which has no serious limitations regarding computational resources. Typical examples of such a setting are the recommendation of IoT apps (e.g., located on some sort of marketplace) and the recommendation of workflows (e.g., located in a workflow repository). Recommendation functionalities that support the task of resource balancing or functionalities supporting the reconfiguration of a gateway installation should be located directly on the gateway in order to be able to perform reconfigurations even in the case that the gateway is not connected to the Internet. Despite limited computational resources available on gateways, recommendations have to be determined in an efficient fashion (Felfernig et al. 2016).

Datasets for evaluation purposes The development of recommendation technologies for IoT scenarios is a rather young discipline and research in the field would strongly profit from the availability of more IoT datasets that enable corresponding tests of, for example, the prediction quality of recommendation algorithms. In the context of end-user development support in IoT scenarios, datasets are helpful that include logs about the development of IoT workflows on remote gateway installations. This information can be exploited to optimize user support, for example, by predicting relevant code-fragments and sensors that should be included.

Distributed data analysis The distributed nature of the Internet of Things and corresponding high amounts of collected data are challenging existing data analysis methods (Stolpe 2016). While approaches to big data analytics (Chen et al. 2015) often follow the paradigm of parallel and high-performance computing, analysis approaches in IoT scenarios are often limited, for example, in terms of bandwidth and energy supply. This is the major motivation for decentralized analysis algorithms that often have to work (partly) on data-generating IoT devices.

[20] movielens.org.
[21] mahout.apache.org.
[22] choco-solver.org.
[23] weevis.org.
[24] choiclaweb.com.

Context-aware recommendation approaches Compared to traditional context-based recommendation approaches, IoT scenarios increase the number of relevant context dimensions (Felfernig et al. 2016). For example, in group-based scenarios (e.g., a group of tourists interested in a city round trip recommendation) example dimensions are not only related to the items to be recommended (e.g., tourist destinations) but also to additional dimensions such as information about potential traffic jams, weather forecasts, occupancy rates of destinations, and availability of public transport (just to mention a few). All these aspects have to be taken into account when building recommendation solutions which also requires the integration of data sources. Recommendation and configuration technologies supporting the ramp-up of IoT infrastructures have to take into account additional aspects such as topological information relevant for the IoT environment (e.g., in the case of animal monitoring applications) and environmental data (e.g., in the context of air pollution monitoring). Such aspects are not relevant in more traditional recommendation and configuration scenarios (Felfernig et al. 2014).

8 Conclusions

In this article, we have provided an overview of existing recommendation approaches besides our proposed recommendation techniques in the Internet of Things (IoT) domain. First, we have given a short overview of existing work related to the application of recommendation technologies in IoT scenarios. Thereafter, we have shown how basic recommendation algorithms can be applied in simple IoT scenarios. Moreover, we have described the challenges that we have faced in the AGILE project. To come over these challenges, we have proposed three recommendation approaches SEQREQ, CONFREQ, and DIAGREQ. SEQREQ provides intelligent workflow/node recommendations whereas CONFREQ and DIAGREQ increases runtime performance and prediction quality of CSP solvers. We have shown how these new approaches can be applied in AGILE project's use cases. After that, we have explained how to select a recommendation approach based on the application domain. Finally, we discussed further research challenges for recommender systems in IoT.

Funding Information Open access funding provided by Graz University of Technology.

References

Adomavicius, G., & Tuzhilin, A. (2015). *Context-aware recommender systems*, (pp. 191–226). Boston: Springer. https://doi.org/10.1007/978-1-4899-7637-6_6.

Amato, F., Mazzeo, A., Moscato, V., Picariello, A. (2013). A recommendation system for browsing of multimedia collections in the internet of things. In Bessis, N., Xhafa, F., Varvarigou, D., Hill, R., Li, M. (Eds.) *Internet of things and inter-cooperative computational technologies for collective intelligence, Studies in Computational Intelligence, vol. 460. Springer*.

Atas, M., Felfernig, A., Erdeniz, S.P., Reiterer, S., Shehadeh, A., Tran, T.N.T. (2017). Cluster-based constraint ordering for direct diagnosis. In *19th international configuration workshop, p. 68*.

Atzori, L., Iera, A., Morabito, G. (2010). The internet of things: A survey. *Computer Networks*, *54*, 2787–2805.

Bahirat, P., He, Y., Menon, A., Knijnenburg, B. (2018). A data-driven approach to developing iot privacy-setting interfaces. In *23rd International Conference on Intelligent User Interfaces* (pp. 165–176). ACM.

Benouaret, I., & Lenne, D. (2015). Personalizing the museum experience through context-aware recommendations. In *2015 IEEE international conference on systems, man, and cybernetics (SMC), pp. 743–748. IEEE*.

Burke, R. (2002). Hybrid recommender systems: Survey and experiments. *UMUAI Journal*, *12*(4), 331–370.

Burke, R.D., Hammond, K.J., Yound, B. (1997). The FindMe approach to assisted browsing. *IEEE Expert*, *12*(4), 32–40.

Cha, S., Ruiz, M., Wachowicz, M., Tran, L., Cao, H., Maduako, I. (2016). The role of an iot platform in the design of real-time recommender systems. In *2016 IEEE 3rd world forum on internet of things (WF-iot), pp. 448-453. Reston, VA, USA*.

Chen, F., Deng, P., Wan, J., Zhang, D., Vasilakos, A.V., Rong, X. (2015). Data mining for the internet of things: Literature review and challenges. *International Journal of Distributed Sensor Networks*, *11*(8), 431,047.

Erdeniz, S.P., Felfernig, A., Atas, M., Tran, T.N.T., Jeran, M., Stettinger, M. (2017). Cluster-specific heuristics for constraint solving. In *International conference on industrial, engineering and other applications of applied intelligent systems, pp. 21–30. Springer*.

Erdeniz, S.P., Maglogiannis, I., Menychtas, A., Felfernig, A., Tran, T.N.T. (2018). Recommender systems for iot enabled m-health applications. In *IFIP International conference on artificial intelligence applications and innovations, pp. 227–237. Springer*.

Falkner, A., Felfernig, A., Haag, A. (2011). Recommendation technologies for configurable products. *AI Magazine*, *32*(3), 99–108.

Felfernig, A., & Burke, R. (2008). Constraint-based recommender systems: Technologies and research issues. In *ACM International conference on electronic commerce (ICEC08), pp. 17-26. Innsbruck, Austria*.

Felfernig, A., Mandl, M., Schippel, S., Schubert, M., Teppan, E. (2010). Adaptive utility-based recommendation. In *International conference on industrial, engineering and other applications of applied intelligent systems, pp. 641–650. Springer*.

Felfernig, A., Hotz, L., Bagley, C., Tiihonen, J. (2014). *Knowledge-based configuration: From research to business cases*, 1st edn. San Mateo: Elsevier/Morgan Kaufmann Publishers.

Felfernig, A., Friedrich, G., Jannach, D., Zanker, M. (2015). Constraint-based recommender systems. In *Recommender systems handbook, pp. 161–190. Springer*.

Felfernig, A., Erdeniz, S.P., Azzoni, P., Jeran, M., Akcay, A., Doukas, C. (2016). Towards configuration technologies for iot gateways. In *International workshop on configuration 2016 (confWS'16), pp. 73-76. Toulouse, France*.

Felfernig, A., Atas, M., Tran, T.N.T., Stettinger, M., Erdeniz, S.P., Leitner, G. (2017). *An analysis of group recommendation heuristics for high- and low-involvement items* (pp. 335–344). Cham: Springer International Publishing. https://doi.org/10.1007/978-3-319-60042-0_39.

Felfernig, A., Boratto, L., Stettinger, M., Tkalčič, M. (2018). *Group recommender systems: an introduction*. Springer.

Finkenzeller, K. (2010). *RFID handbook: Fundamentals and applications in contactless smart cards, radio frequency identification and near-field communication*. New York: Wiley.

Greengard, S. (2015). *The internet of things*. Cambridge: MIT Press.

Han, J., Pei, J., Mortazavi-Asl, B., Pinto, H., Chen, Q., Dayal, U., Hsu, M. (2001). Prefixspan: Mining sequential patterns efficiently by prefix-projected pattern growth. In *Proceedings of the 17th international conference on data engineering, pp. 215–224*.

Jannach, D., Zanker, M., Felfernig, A., Friedrich, G. (2010). *Recommender systems – an introduction*. Cambridge: Cambridge University Press.

Konstan, J., Miller, B., Maltz, D., Herlocker, J., Gordon, L., Riedl, J. (1997). Grouplens: Applying collaborative filtering to usenet news full text. *Commission of the ACM*, *40*(3), 77–87.

Koren, Y., Bell, R., Volinsky, C. (2009). Matrix factorization techniques for recommender systems. *IEEE Computer*, *42*(8), 30–37.

Lee, J.S., Su, Y.W., Shen, C.C. (2007). A comparative study of wireless protocols: bluetooth, uwb, zigbee, and wi-fi. In *33rd annual conference of the industrial electronics society, IECON 2007. IEEE 2007, pp. 46–51. IEEE*.

Leitner, G., Felfernig, A., Fercher, A., Hitz, M. (2014). Disseminating ambient assisted living in the rural area. *Sensors*, *14*(8), 13,496–13,531.

Magerkurth, C., Sperner, K., Meyer, S., Strohbach, M. (2011). Towards context-aware retail environments: An infrastructure perspective. In *MobileHCI 2011, pp. 1–4. Stockholm, Sweden*.

Maglogiannis, I., Ioannou, C., Tsanakas, P. (2016). Fall detection and activity identification using wearable and hand-held devices. *Integrated Computer-Aided Engineering*, *23*(2), 161–172.

Martin, P., Ho, B.J., Grupen, N., Munoz, S., Srivastava, M. (2014). An ibeacon primer for indoor localization: Demo abstract. In *Proceedings of the 1st ACM Conference on Embedded Systems for Energy-Efficient Buildings, pp. 190–191. ACM*.

Martino, S.D., & Rossi, S. (2016). An architecture for a mobility recommender system in smart cities. *Procedia Computer Science*, *98*, 425–430.

Mashal, I., Alsaryrah, O., Chung, T.Y. (2016). Performance evaluation of recommendation algorithms on internet of things services. *Physica A*, *451*, 646–656.

Masthoff, J. (2011). Group recommender systems, Recommender Systems Handbook, pp. 677–702.

Menychtas, A., Tsanakas, P., Maglogiannis, I. (2016). Automated integration of wireless biosignal collection devices for patient-centred decision-making in point-of-care systems. *Healthcare Technology Letters*, *3*(1), 34–40.

Munoz-Organero, M., Ramirez-Gonzalez, G., Munoz-Merino, P., Loos, C. (2010). A collaborative recommender system based on space-time similarities. *IEEE Pervasice Computing*, *9*(3), 81–87.

Pazzani, M., & Billsus, D. (1997). Learning and revising user profiles: The identification of interesting web sites. *Machine Learning*, *27*, 313–331.

Prud'homme, C., Fages, J.G., Lorca, X. (2016). Choco solver documentation. TASC, INRIA Rennes, LINA CNRS UMR 6241.

Ray, P. (2015). Generic internet of things architecture for smart sports. In *International conference on control, instrumentation, and communication technologies (ICCICCT), pp. 405–410*.

Srikant, R., & Agrawal, R. (1996). Mining sequential patterns: Generalizations and performance improvements. Advances in Database Technology—EDBT'96, pp. 1–17.

Stolpe, M. (2016). The internet of things: Opportunities and challenges for distributed data analysis. *ACM SIGKDD Exlorations Newsletter*, *18*, 15–34.

Tsang, E. (1993). *Foundations of constraint satisfaction*. New York: Academic Press.

Valtolina, S., Mesiti, M., Barricelli, B. (2014). User-centered recommendation services in internet of things era. In *CoPDA2014 workshop. Como, Italy*.

Venturini, G. (1993). Sia: a supervised inductive algorithm with genetic search for learning attributes based concepts. In *European conference on machine learning, pp. 280–296. Springer*.

Visalakshi, N.K., & Thangavel, K. (2009). Impact of normalization in distributed k-means clustering. *International Journal of Soft Computing*, *4*(4), 168–172.

Winterfeldt, D., & Edwards, W. (1986). *Decision analysis and behavioral research*. Cambridge: Cambridge University Press.

Yavari, A., Jayaraman, P.P., Georgakopoulo, D. (2016). Contextualised service delivery in the internet of things. In *2016 IEEE 3Rd world forum on internet of things (WF-iot), pp. 454-459. Reston, VA, USA*.

Zaki, M.J. (2001). Spade: an efficient algorithm for mining frequent sequences. *Machine learning*, *42*(1), 31–60.

Affiliations

Alexander Felfernig[1] · Seda Polat-Erdeniz[1] · Christoph Uran[1] · Stefan Reiterer[1] · Muesluem Atas[1] · Thi Ngoc Trang Tran[1] · Paolo Azzoni[2] · Csaba Kiraly[3] · Koustabh Dolui[3]

Alexander Felfernig
alexander.felfernig@ist.tugraz.at

Christoph Uran
uran@ist.tugraz.at

Stefan Reiterer
reiterer@ist.tugraz.at

Muesluem Atas
muatas@ist.tugraz.at

Thi Ngoc Trang Tran
ttrang@ist.tugraz.at

Paolo Azzoni
paolo.azzoni@eurotech.com

Csaba Kiraly
kiraly@fbk.eu

Koustabh Dolui
k.dolui@fbk.eu

[1] Institute for Software Technology, Inffeldgasse 16B/2, 8010 Graz, Austria

[2] Eurotech Group, I-33020 Amaro, Italy

[3] FBK CREATE-NET, via alla Cascata 56/D, 38123 Trento, Italy