Towards Context-Aware Task Recommendation

Chuong Cong Vo, Torab Torabi and Seng W. Loke
Department of Computer Science & Computer Engineering, La Trobe University
{*c.vo, t.torabi, s.loke*}*@latrobe.edu.au*

***Abstract*—Pervasive computing environments can provide a diverse range of computing capabilities. But they are overwhelming for us as we attempt to accomplish our daily tasks. We need to know what resources are at hand and what tasks a PCE supports. Our ongoing research aims to minimise this overload by proposing a context-aware task recommender system named TASKREC which can recommend or automatically accomplish feasible and relevant tasks for users according to user's situation and environment capabilities. This paper formalises the problem of task recommendation, presents a system architecture for TASKREC, and an early prototype implementation. Finally, we suggest open opportunities for future research on adaptation techniques for context-aware systems.**

***Index Terms*—Recommender Systems, Task Recommendation, Pervasive Computing, Context-Awareness, Task Computing**

I. INTRODUCTION

The world is moving towards universally connected spaces where "technologies weave themselves into the fabric of everyday life. . . " [1]. Spaces in such a world are often referred to as pervasive (or ubiquitous) computing environments (PCEs) [2]. However, a challenge of adding intelligent technologies to our lives is to "support our activities, complement our skills, and add to our pleasure, convenience, and accomplishments, but not to our stress" [3].

Adding technologies into everyday settings, on one hand, facilitates our activities but on the other hand, overwhelms our perception and cognition ability by the overload of information, services, and configurations [4], [5]. To exploit such an environment, users must (1) recognise capabilities of the environment to issue feasible tasks; (2) map their high-level goals of tasks to low-level operational vocabularies; and (3) properly specify constraints for tasks subject to contextual information and available capabilities. These requirements may be beyond ordinary users as complexity, diversity, and sheer number of devices (as well as different combinations of ways they can work together) continually rises.

Invisibility is a goal of PCEs, where devices should operate naturally and unobtrusively, and blend well into the environment. However, the invisibility can pose problems. People may not recognise the presence of facilities available to them. For example, there could be thirty to a hundred computing devices inside a future meeting room. They may provide various capabilities (some capabilities perhaps provided by a combination of such devices), some of which the user may not even be aware and some of these devices may not be visible to him/her.

To address the overload and invisibility problems, there have been research efforts on *ubiquitous recommendation systems* [6] and *context-aware recommender systems* [7] that aim to recommend relevant information and services to users. However, users are still required to integrate the recommended information and services to achieve their intended tasks [8]. This calls for *context-aware task recommender systems* which can recommend or even automatically accomplish relevant tasks for users subject to current context and available services. In other words, the context-aware task recommender system can deal with the question of "*What tasks should I do with devices, information, and services I currently have?*"

This paper presents a context-aware task recommender system (TASKREC). Derived from the behavioural psychology science which asserts *people behave similarly in similar situations* [9], we assume that *people accomplish similar tasks in similar situations*. Our task recommendation applies multiple strategies: *collaborative filtering based on situation similarity*, *knowledge-based filtering*, and *utility-based filtering*.

The paper is organised as follows. Section II presents a motivating scenario for which TASKREC is useful. Section III formulates the task recommendation problem. Strategies for recommendation are elaborated in Section IV. Section V presents an architecture for TASKREC. The implementation and simulation of the scenario are described in Section VI. Section VII outlines the related work. Finally, a conclusion and an ongoing research agenda are given in Section VIII.

II. MOTIVATING SCENARIO

This section describes applications of TASKREC via a scenario involving places: **`house`**, **`car park`**, and **`office`**.

"At 7am on a cold, rainy Monday morning, when Bob has stepped out towards his office, TASKREC *on his smartphone recommends him a task '**`Drive to work`**'. As Bob selects this task, the heater and the TV are turned off; his smartphone is switched to outdoor mode; all calls to his home phone will now be forwarded to his smartphone; the doors of his house are closed. The car's radio is turned to news channel while he is driving. . .*

When Bob is about to approach the car park at the university, TASKREC *recommends '**`Find a parking place`**' task which guides him to an available parking spot. The car is parked, he walks to his office. The door is opened as he is nearly in the font of it. The room's overhead lights and the heater are turned on and adjusted to his preference as he steps into the room. His smartphone is now changed to quiet mode. The teapot is on because he always has a cup of tea in early mornings before focusing on the work. . . "*

III. Problem Formalisation

A. Ideas and Assumptions

The main questions are: what tasks to be recommended; how to determine them; and in what sequence they should appear. To answer these questions, we make the following assumptions.

(i) User always prefers tasks which are highly *relevant* to his current situation. A task is called relevant to the user if accomplishing the task will satisfy the user's intention.
(ii) For tasks having the same degree of relevance, the more *feasible* the task is, the more preferred it is. The feasibility for accomplishing a task depends on the current capabilities of the environment.
(iii) If two tasks are equally feasible, then the task which is more *autonomous* is more preferred. The autonomy of task operation expresses user's intervention required to accomplish the task (without concern about the capability of the environment).

Accordingly, the problem of task recommendation now becomes the problem of ranking tasks based on three measures: the relevance of task to user's situation, the feasibility to accomplish a task in the current environment, and the autonomy of task performance. In the following, we define concepts used in our methodology for task recommendation and how these measures are computed.

B. Tasks

A task is a unit of work. Tasks are classified to places and devices which are called *place-related* and *device-related tasks*. For example, *room-related tasks* are **`Make room bright`**, **`Make room warm`**, and **`Make tea`**. The list of all potential tasks in a given environment is called a *task repository*.

Fulfillment of a task may require capabilities from the environment in which the task will be carried out. For example, the task of **`Make room warm`** requires capabilities for increasing temperature which can be provided by, e.g., **`heater-like devices`**. We call them *required capabilities*.

A description of how a task should be executed is called *task flow*. A task flow decomposes a task into *sub-tasks*. The sub-tasks are then decomposed further until they can be executed directly by invoking services.

To measure how the operation of a task being intrusive to a user, we introduce the concept of *task autonomy*. The autonomy of a task indicates to what extent the task can be accomplished automatically and how much user intervention is required. Obviously, the degree of autonomy depends on the autonomy of sub-tasks of a task.

For measuring the feasibility of a task, we take into account the capabilities of the environment which refers to software services and device functionalities. The PCEs are highly dynamic environments, hence, it is significant to measure the feasibility of tasks for recommendation.

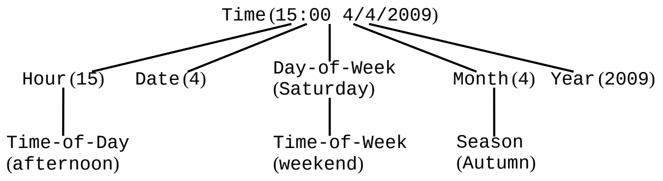

Fig. 1. The decomposition of the time context.

C. Context and Situation

We adopt an operational definition of *context* by Dey *et al.* [10]: "*Context is any information that can be used to characterise the situation of an entity.*"

Ontology-based context modelling [11] is widely accepted thank to semantic expressiveness and knowledge sharing of ontologies. As designers, we model contexts as *attribute-value pairs*, e.g. (**`Time-Context`**, '**`15:00 4/4/2009`**'). Each context attribute can be decomposed into different levels of generalisation (see **`Time-Context`** in Fig. 1). The system can query for values at arbitrary levels of generalisation. The reasoning is done by the context manager.

Situations are characterised by contextual information. We argue that for any matters a situation is modelled and represented (e.g., *formal logics* [12]), we are able to convert the models into the *vector-based situation model*. Formally, let s^t be a situation at time t, we define s^t as a vector of contexts at time t: $\mathbf{s}^t = (\mathbf{c}_1^t, \mathbf{c}_2^t, \ldots, \mathbf{c}_n^t)$, where

- n is the number of context attributes;
- $\mathbf{c}_i^t$ is value of context attribute i $(i \in 1..n)$ at time t;
- $\mathcal{C}_i$ is the set of possible values of context attribute i;
- $\mathcal{S} = \mathcal{C}_1 \times \ldots \times \mathcal{C}_n$ denotes space of possible situations.

In this definition, for simplicity, we use $\mathbf{c}_i^t$ to represent a context attribute-value pair. The index i represents for the name of a context attribute. For example, given $\mathcal{S} =$ **`Day-Of-Week`** $\times$ **`Place-Type`**, then $\mathrm{s} =$ ('**`Monday`**', '**`Library`**') is a possible situation. Note that a context value can be a single number (e.g., **`Temperature`**) or a structured object (e.g., **`Location`**, **`Time`**). A special value for all context attributes is **`null`** value. For a given situation, context attributes with values of **`null`** will be treated as unknown or insignificant for characterising the situation.

IV. Strategies for Recommendation

To find tasks relevant to an active user in his current situation, we apply a combination of collaborative filtering, knowledge-based, and utility-based filtering.

A. Collaborative Filtering Based on Situation Similarity

The tasks similar to tasks previously accomplished by *similar users* in *similar situations* are considered relevant to the active user. In what follows, we will introduce the measures for *task similarity*, *situation similarity*, and *user similarity*.

1) Task Similarity: Similarity between two tasks is identified based on their effects on the situation. For example, **`Open windows`** and **`Turn overhead lights on`** are similar as they are both for increasing the brightness of a room. Formally, we denote the similarity between two tasks $\mathbf{t}$ and $\mathbf{t}'$ as $TS(\mathbf{t}, \mathbf{t}')$.

TABLE I
AN EXAMPLE OF TASK-BASED SITUATION SIMILARITY.

	$\mathbf{t}'_1$	$\mathbf{t}'_2$
$\mathbf{t}_1$	0.7	0.6
$\mathbf{t}_2$	0.6	1.0
$\mathbf{t}_3$	0.1	0.9

2) Situation Similarity: Pure similarity between situations is inferred from the similarity of their local values of context attributes. In our system, this measure is used to find previous situations similar to a newly unknown situation. Given two situations $\mathbf{s} = (\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_n)$ and $\mathbf{s}' = (\mathbf{c}'_1, \mathbf{c}'_2, \ldots, \mathbf{c}'_n)$ (note that all situations are defined using the same context attributes and the orders are significant), the pure similarity between $\mathbf{s}$ and $\mathbf{s}'$, $SS_{\mathrm{P}}(\mathbf{s}, \mathbf{s}')$, is defined by

$$SS_{\mathrm{P}}(\mathbf{s}, \mathbf{s}') = \sum_{i=1}^{n} w_i * sim_i(\mathbf{c}_i, \mathbf{c}'_i), \quad (1)$$

where $w_i \in [0, 1]$ $(\sum_{i=1}^{n} w_i = 1)$ denotes the significance weight assigned to the context attribute i; $sim_i(\mathbf{c}_i, \mathbf{c}'_i)$ is the *per-context attribute similarity* between two values $\mathbf{c}_i$ and $\mathbf{c}'_i$ for the context attribute i. Many techniques can be used to calculate the per-context attribute similarity (e.g., [13], [14]).

A limitation of the pure situation similarity is the need for explicitly specifying significance weights assigned to context attributes (i.e., w_i) while there is no unified method to determine such the weights. Therefore, we propose a new measure for computing similarity between situations called *task-based situation similarity*. Specifically, two situations are similar if the tasks typically accomplished in these situations are similar. For example, the situation of "**`In-Meeting`**" and "**`In-Theatre`**" are similar with respect to the task of "**`Change mobile to quiet mode`**" because people usually change their mobile to quiet mode when they are in these situations.

Formally, let $(\mathbf{t}_i), i \in 1..k$, and $(\mathbf{t}'_j), j \in 1..l$, be tasks which are accomplished in $\mathbf{s}$ and $\mathbf{s}'$, respectively, the task-based similarity between $\mathbf{s}$ and $\mathbf{s}'$, $SS_{\mathrm{T}}(\mathbf{s}, \mathbf{s}')$, is defined by

$$SS_{\mathrm{T}}(\mathbf{s}, \mathbf{s}') = \frac{\sum_{i=1}^{k} \sum_{j=1}^{l} TS(\mathbf{t}_i, \mathbf{t}'_j)}{k * l}. \quad (2)$$

For example, there are three tasks accomplished in $\mathbf{s}$ and two in $\mathbf{s}'$. The mutual similarities of these tasks are given in Table I. By applying (2), we have $SS_{\mathrm{T}}(\mathbf{s}, \mathbf{s}') = 0.65$.

3) User Similarity: Pure similarity between users is the sum of the similarity of their characteristics such as *age*, *gender*, and *occupation*. This measure is used to find previous users similar to the active user who is new to the system. For measuring similarity between known users, we propose *task-based user similarity*. Particularly, the similarity between two users is calculated based on similarities between tasks they have previously accomplished in similar situations. Formally, let $\mathbf{u}$ and $\mathbf{u}'$ be two users in consideration; let $(\mathbf{s}_i), i \in 1..k$, and $(\mathbf{s}'_j), j \in 1..l$, be situations in which, respectively, $\mathbf{u}$ and $\mathbf{u}'$ have experienced and accomplished some tasks. The similarity between $\mathbf{u}$ and $\mathbf{u}'$ is defined by

$$US_{\mathrm{T}}(\mathbf{u}, \mathbf{u}') = \frac{\sum_{i=1}^{k} \sum_{j=1}^{l} SS_{\mathrm{T}}(\mathbf{s}_i, \mathbf{s}'_j)}{k * l}. \quad (3)$$

B. Knowledge-Based Filtering

It is observed that individual tasks are often associated with context (e.g., *places* and *devices*). Therefore, context may determine potential tasks. We propose to construct *task repositories* oriented to *places* and *devices* as previously discussed in Section III-B.

Task frequency, *task sequences*, *task groups*, and *task hierarchies* are good sources for prediction of next tasks. The knowledge-based filtering enables us to overcome the problem of new users and new tasks which is an inherent issue of the collaborative filtering method.

C. Utility-Based Filtering

1) Task Feasibility: The feasibility of a task (called *task feasibility*) defines the degree of feasibility to accomplish the task in a given environment. It is calculated by matching *required capabilities* of the task with *provided capabilities* of the environment.

2) Task Autonomy: The autonomy of a task (*task autonomy*) indicates to what extent the task must be accomplished by the environment. We call $a_1, a_2, \ldots, a_m$ (where m is the number of sub-tasks within the taskflow) the autonomy of the sub-tasks, then the autonomy of the task t is $\sum_{i=1}^{m} a_i/m$.

V. SYSTEM COMPONENTS

The system has five main components. *Context Manager* is for management of contextual information. *Resource Manager* is responsible for discovery and management of available devices and services in the environment. *Task Execution Manager* is to execute selected tasks and manage their executions. *Task Recommendation Engine* (TRE) is our core component. It use knowledge about environment capabilities, contextual information, and history data provided by Context Manager and Resource Manager for reasoning about the relevances of tasks. The final is TASKREC *Clients* running on smart devices (e.g., PDA, smartphone) which continuously listens to TRE for receiving recommendation.

VI. IMPLEMENTATION

We illustrate the operation of TASKREC via a *Context Simulator*. We use *Socket Communication* technology for exchange messages between TRE and TASKREC Client. The components of TASKREC are written in *J2SE* while TASKREC Client is written using *Java ME* and deployed on *S40 Platform Nokia Emulator*.

Consider a situation: "*At 8am on a cold, rainy Monday morning, ... when Bob steps into his office*". By applying the knowledge-based filtering, the system can reduce the universe of tasks (perhaps hundreds of tasks) to *Office-related tasks* and *Mobilephone-related tasks*.

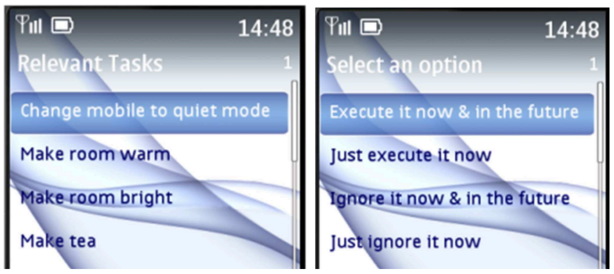

Fig. 2. Reduced list of task recommended on the user's mobile device.

Next, by applying situation similarity-based collaborative filtering, the list is reduced to 4 tasks. These tasks are then ranked based on their feasibility and autonomy as shown on the left image of Fig. 2. The user can specify his preference of how each task can be recommended in the future using options as given in the right image of Fig. 2.

VII. Related Work

There has been research on task computing [4], [5], [8]. However, few have focused on task recommendation in PCEs.

Cheng *et al*. [15] propose a system which compares past situations with the current situation to find similar situations. Then, applications typically performed in these similar situations are ranked and recommended. This approach actually recommends single applications while our approach is to recommend tasks which would require multiple applications and services for accomplishment.

Messer *et al*. [16] present a middleware called *InterPlay* for seamless device integration and task orchestration in a networked home. It asks users to express their tasks via a pseudo-English interface and assumes that the users have knowledge about feasible tasks. In contrast, our approach can recommend relevant, feasible tasks without these requirements.

Ni *et al*. [17] introduce an algorithm for discovering *active* tasks. The algorithm matches the current context of a user with required context of tasks. Their solution can discover feasible tasks but potentially irrelevant tasks. Fukazawa *et al*. [18] propose a system which uses object names specified by users (e.g., *cafe shop*, *theatre*) for retrieving tasks which are associated with these names. Our approach has integrated this knowledge into *place-related task repository*.

The *Homebird* system [19] automatically discovers features of other devices in the surrounding and suggests the user certain tasks that can be performed together with those devices. Because this approach does not consider user situation, it can recommend feasible tasks which may be not relevant.

VIII. Conclusion and Future Work

This paper introduced the problem of context-aware task recommendation in PCEs. Our solution for task recommendation uses collaborative filtering and knowledge-based filtering together with reasoning about task feasibility and task autonomy. We proposed a new measure for task-based situation similarity. We presented the architecture and an implementation for the task recommender system called TaskRec.

The next step should evaluate the system via user studies. Future research will also investigate potential benefits from applying task recommendation as an adaptation technique for context-aware systems. Another issue is to use learning techniques and data mining for computing importance weights assigned to context attributes towards individual tasks. Furthermore, the conflicts of recommendations in multiuser environments bring up challenges that we are also aware of.

Acknowledgment

The research was carried out whilst the first author was supported by a La Trobe University Tuition Fee Remission and Postgraduate Research Scholarship.

References

[1] M. Weiser, "The computer for the 21st century," *Scientific American*, vol. 3, no. 265, pp. 94–104, 1991.
[2] M. Satyanarayanan, "Pervasive computing: vision and challenges," *Personal Communications, IEEE*, vol. 8, no. 4, pp. 10–17, 2001.
[3] D. A. Norman, *The Design of Future Things*. Basic Books, 2007.
[4] Z. Wang and D. Garlan, "Task-driven computing," School of Computer Science, Carnegie Mellon University, Tech. Rep., 2000.
[5] D. Garlan, D. Siewiorek, A. Smailagic, and P. Steenkiste, "Project Aura: toward distraction-free pervasive computing," *Pervasive Computing, IEEE*, vol. 1, no. 2, pp. 22–31, Apr-Jun 2002.
[6] D. W. McDonald, "Ubiquitous recommendation systems," *Computer*, vol. 36, no. 10, pp. 111–112, 2003.
[7] G. Adomavicius, R. Sankaranarayanan, S. Sen, and A. Tuzhilin, "Incorporating contextual information in recommender systems using a multidimensional approach," *ACM Trans. Inf. Syst.*, vol. 23, no. 1, pp. 103–145, 2005.
[8] R. Masuoka, B. Parsia, and Y. Labrou, "Task computing – the semantic web meets pervasive computing," *The SemanticWeb - ISWC 2003*, pp. 866–881, 2003.
[9] D. Magnusson and B. Ekehammar, "Similar situations–similar behaviors? a study of the intraindividual congruence between situation perception and situation reactions," *Journal of Research in Personality*, vol. 12, pp. 41–48, 1978.
[10] A. K. Dey, "Understanding and using context," *Personal and Ubiquitous Computing*, vol. 5, no. 1, pp. 4–7, Feb. 2001.
[11] H. Chen, T. Finin, and A. Joshi, "An ontology for context-aware pervasive computing environments," *Knowl. Eng. Rev.*, vol. 18, no. 3, pp. 197–207, 2003.
[12] J. Barwise and J. Perry, "Situations and attitudes," *The Journal of Philosophy*, vol. 78, no. 11, pp. 668–691, 1981.
[13] S. W. Loke, "Representing and reasoning with situations for context-aware pervasive computing: a logic programming perspective," *Knowl. Eng. Rev.*, vol. 19, no. 3, pp. 213–233, 2004.
[14] C. B. Anagnostopoulos, Y. Ntarladimas, and S. Hadjiefthymiades, "Reasoning about situation similarity," in *3rd International IEEE Conference on Intelligent Systems*, Sept. 2006, pp. 109 –114.
[15] D. Cheng, H. Song, H. Cho, S. Jeong, S. Kalasapur, and A. Messer, "Mobile situation-aware task recommendation application," in *The Second International Conference on Next Generation Mobile Applications, Services, and Technologies*, 2008.
[16] A. Messer, A. Kunjithapatham, M. Sheshagiri, H. Song, P. Kumar, P. Nguyen, and K. H. Yi, "InterPlay: A middleware for seamless device integration and task orchestration in a networked home," in *PERCOM'06*. IEEE Computer Society, 2006, pp. 296–307.
[17] H. Ni, X. Zhou, D. Zhang, and N. Heng, "Context-dependent task computing in pervasive environment," *Ubiquitous Computing Systems*, pp. 119–128, 2006.
[18] Y. Fukazawa, T. Naganuma, K. Fujii, and S. Kurakake, "A framework for task retrieval in task-oriented service navigation system," in *OTM Workshops 2005*. Springer Berlin/Heidelberg, 2005, pp. 876–885.
[19] O. Rantapuska and M. Lähteenmäki, "Homebird–task-based user experience for home networks and smart spaces," in *PERMID 2008*, 2008.