Things of Interest Recommendation by Leveraging Heterogeneous Relations in the Internet of Things

LINA YAO, The University of New South Wales, Australia
QUAN Z. SHENG, The University of Adelaide, Australia
ANNE H. H. NGU, The Texas State University, USA
XUE LI, The University of Queensland, Australia

The emerging Internet of Things (IoT) bridges the gap between the physical and the digital worlds, which enables a deeper understanding of user preferences and behaviors. The rich interactions and relations between users and things call for effective and efficient recommendation approaches to better meet users' interests and needs. In this article, we focus on the problem of things recommendation in IoT, which is important for many applications such as e-Commerce and health care. We discuss the new properties of recommending things of interest in IoT, and propose a unified probabilistic factor based framework by fusing relations across heterogeneous entities of IoT, for example, user-thing relations, user-user relations, and thing-thing relations, to make more accurate recommendations. Specifically, we develop a hypergraph to model things' spatiotemporal correlations, on top of which implicit things correlations can be generated. We have built an IoT testbed to validate our approach and the experimental results demonstrate its feasibility and effectiveness.

Categories and Subject Descriptors: H.3.3 [**Information Search and Retrieval**]: Query Formulation, Selection Process; H.3.5 [**Online Information Services**]: Web-based Services

General Terms: Design, Algorithms, Performance

Additional Key Words and Phrases: Internet of things, data mining, hypergraph, latent relationships, recommendation

ACM Reference Format:
Lina Yao, Quan Z. Sheng, Anne H. H. Ngu, and Xue Li. 2016. Things of interest recommendation by leveraging heterogeneous relations in the Internet of Things. ACM Trans. Internet Technol. 16, 2, Article 9 (March 2016), 25 pages.
DOI: http://dx.doi.org/10.1145/2837024

1. INTRODUCTION

The emerging Internet of Things (IoT) is accelerating the growth of data available on the Internet, which makes the traditional search paradigms incapable of digging the information that people need from massive and deep resources. Furthermore, given the dynamic nature of devices, organizations, and social structures involved in the

Quan Z. Sheng's research has been partially supported by Australian Research Council (ARC) Future Fellowship FT140101247 and Discovery Project Grant DP140100104. Lina Yao's research has been supported by ARC Discovery Early Career Researcher Award DE160100509.
Authors' addresses: L. Yao, School of Computer Science and Engineering, the University of New South Wales, NSW 2052, Australia; email: lina.yao@cse.unsw.edu.au; Q. Z. Sheng, School of Computer Science, the University of Adelaide, SA 5005, Australia; email: michael.sheng@adelaide.edu.au; A. H. H. Ngu, Department of Computer Science, Texas State University, TX 78666-4616, USA; email: angu@txstate.edu; X. Li, School of ITEE, the University of Queensland, Queensland, 4072, Australia; email: xueli@itee.uq.edu.au.

DOI: http://dx.doi.org/10.1145/2837024

IoT, intelligent and automated approaches are needed to support decision makers with the knowledge derived from the vast amount of information available through IoT networks. IoT is more desirable of an effective and efficient paradigm of proactive *discovering* rather than postactive searching, in which a user who has certain information needs will search on the Internet for related resources, while the concept of "discovery" refers to the ability to push relevant and related resources to the user, especially according to the metadata collected from the user's historical activities in terms of activities on resources or with other people.

To realize proactive discovery on the future Internet, in this article, we explore the topic of *things recommendation*. Things recommendation is a crucial step for promoting and taking full advantage of the IoT, where it benefits the individuals, businesses, and society on a daily basis on two main aspects. On the one hand, it can deliver relevant things (e.g., things that users might need) to users based on their preferences and interests. On the other hand, it can also serve to optimize the time and cost of using IoT in a particular situation. Physical things in reality have multiple unique attributes. For instance, physical things have states (e.g., in use or not in use; expired or not expired). When a certain thing is in use, it cannot be used simultaneously by another user. Under this circumstance, a recommender system can refer the user to a list of similar things that are available. Recommendation in IoT is more spontaneous, context dependence and monopolistic under certain circumstances compared with traditional recommendations. Consequently, things recommendation is much more complex than traditional recommender systems like recommending books or movies to consumers. The following are some of the reasons that cause the additional complexity in things recommendation:

—*Diverse relations.* The heterogeneous relationships and interdependencies between entities (people and things), data, information, and knowledge in the IoT are growing exponentially, leading to complexity and gross inefficiencies. Failure to distinguish the various types of relations may result in inaccurate recommendation results.
—*Highly dynamic.* The dynamic nature of things calls for models that can rapidly adapt to the constantly changing status of things and always present the most up to date recommendation results.
—*Spatiotemporal correlated.* User behaviors on things are intrinsically correlated both spatially and temporally. The heterogeneous nature of spatiotemporal data is a big challenge for recommendation.
—*Incomplete description.* Things are usually associated with descriptions, categories ,or social tags to describe their attributes of functionality or nonfunctionality. However, unlike traditional recommendation (i.e., recommending a movie), the textual information associated with things is usually incomplete and ambiguous.

In light of the preceding challenges, we propose a probabilistic matrix factorization based framework to address the things recommendation problem in IoT. We fuse information from social networks of users and correlation networks of things, by learning shared latent factors stemming from the probabilistic matrix factorization on three matrices, namely, *users relationships*, *things' correlations*, and *observable things' usage interactions*. A user's social connections can provide valuable clues about her preferences. However, constructing a collaborative user model is not trivial, since not all information from the social environment is equally useful. In our framework, we leverage the social relation information in designing the social regularization term to constrain the matrix factorization objective function. The social regularization term can indirectly model the propagation of the user's interest and preference.

We also propose a graph-based approach to uncover things' correlations by analyzing their usage frequency, which reflects two implications: (i) users' preferences or needs on

things can be partially revealed by how frequently a user uses a thing; and (ii) higher usage frequency on a thing from a user indicates more needs on this thing. Our approach adopts the probabilistic matrix factorization based on dimensionality reduction techniques. More specifically, the user-thing frequency matrix (induced from things usage logs) is factorized into two low-dimensional factor matrices: *user latent matrix* and *thing latent matrix*. Finally, the state of physical things are rapidly updated and one thing cannot be used by multiple users in certain circumstances. For instance, an ATM machine being in use can not be used by another person. We embed a sequential Monte Carlo based mechanism into our proposed framework to continuously refine the things availability temporally, and integrate this latest information in the recommendation results. Our main contributions are summarized as follows:

—We focus on things recommendation and systematically analyze things' correlations, which has substantial impact on users behaviors on things usage and human decision making process. This work is based on our previous work on correlation discovery of ubiquitous things in IoT and our preliminary exploration on things recommendation [Yao et al. 2013, 2014b].
—Things correlations can be uncovered and utilized for serving user-thing usage pattern prediction. In particular, we develop a hypergraph-based approach to derive things implicit correlations in a query-rank style, in which complex relations in user-thing interactions can be represented at utmost without unnecessary information loss.
—We propose a sequential Monte Carlo based method to continuously track the availability status of physical things, and embed such dynamic feature of physical things into our proposed framework to improve recommendation results.
—We validate and evaluate our model on a real-world IoT testbed. The experimental results demonstrate the feasibility and effectiveness of our approach.

The rest of this article is organized as follows. We formulate our problem in Section 2. We present the proposed framework and technical details in Section 3. We describe the setting of our experimental environment and report our evaluation results in Section 4. Finally, we overview the related work in Section 5 and discuss some future research directions in Section 6.

2. PROBLEM FORMULATION

We argue that both users' relations and things' correlations should be taken into account for making accurate recommendations in IoT. Indeed, users' relations (e.g., friendships) can have significant impact on things usage patterns. Personal tastes are correlated. Research in Kameda et al. [1997] shows that friendships and relations between users play a substantial role in human decision making in social networks. For instance, people usually turn to a friend's advice about a commodity (e.g., hair straightener) or a restaurant before they go for them. Sometimes, such influence from the friend circle is even more substantial than high ratings given by other people [Sinha and Swearingen 2001]. In addition, if one user is using a certain thing (e.g., a laptop), the status of thing can spread to her social network circle, and the people in her circle will know the thing is not available. They therefore need to turn to other similar things if they need them. Such kind of situation typically happens in the IoT, and affects the things' usage events. It also affects the people's decision making process and recommendation results.

As mentioned previously, things are functionality oriented and things that have similar or same functionalities hold strong relationships. We argue that physical things have more distinctive structures and connections in terms of their functionalities in real life (i.e., usefulness), as well as nonfunctionalities (i.e., availability). For example,

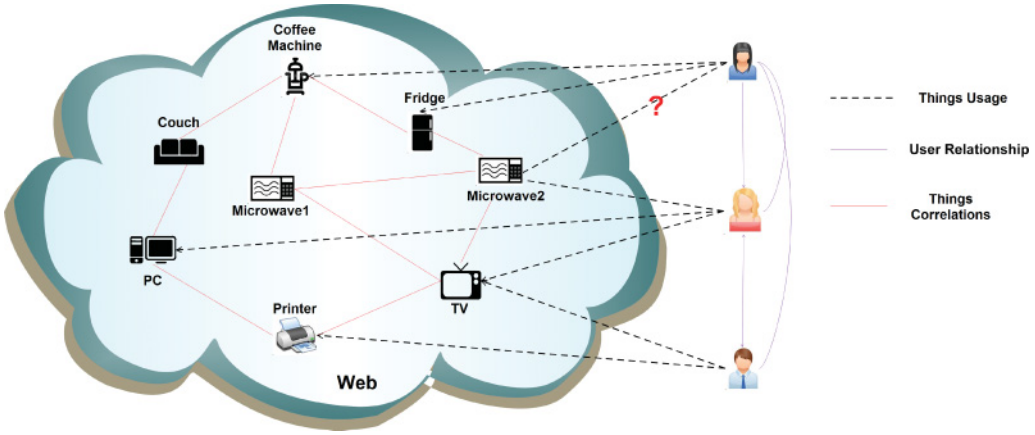


Fig. 1. The graph induced from users' social networks and things correlations. The connections consist of three types of links, including users' relationships within *users' social network*, things correlations with correlation graph of things, and things usage links within *user-thing interactions*. Our task is to predict the dyadic relationship between users and things, in other words, recommending certain things to users.

different things provide different functionalities (e.g., microwave and printer), and will be of interest to different groups of people. Pairwise things with strong correlations indicate either they have similar functionalities or they have more likelihood to be used together. For instance, a water tap and a chop board are both in use when we prepare meals, since most of the time we need to wash cooking materials (e.g., vegetables) before chopping them. In either case, similar things usage pattern will be reflected and the latent correlations between things will be revealed, which have the potential to affect the recommendation process in an implicit way and have not been explored much yet.

Figure 1 gives an overview of user-thing interactions in the IoT, in which the addition of interactions (things usage events) creates new relations and correlations. We can build three graphs from it: a *user-thing* graph, a *thing-thing* graph, and a *user-user* graph.

—User-thing graph. In the user-thing graph (shown in the middle with black dash lines of Figure 1), there are two types of nodes, users and things. An edge starting from a user and ending at a thing indicates that the user has used or invoked the thing (e.g., used an ATM machine in KFC), and the weight of the edge can indicate the number of usage. We name this as *things usage events* in this article.
—*User-user graph*. In the user-user graph (shown on the right with blue lines in Figure 1), a node is a user and an edge is the original connection between two users in an existing social network.
—*Thing-thing graph*. In the thing-thing graph (shown on the left with red lines in Figure 1), a node is a thing and a directed edge between two things indicates that some users consecutively use and invoke the things. The weight associated with an edge represents the strength of the correlation between two things, which is derived from the users' interactions on the things; for example, two things may be connected if they have been used at a similar time or at similar types of locations. These implicit relations initially cannot be observed and are inferred from things usage events (Section 3.2).

Formally, given the historical things' usage logs $Y \in \mathbb{R}^{I \times J}$ of I users $\mathcal{P} = \{p_1, \ldots, p_I\}$ and J things $\mathcal{T} = \{t_1, \ldots, t_J\}$ with r_{ij} as the number of times user p_i used thing t_j. We have the profile information for each y_{ij}, such as temporal information, which indicates

what time the usage event happens, and spatial information, which indicates where the thing is used. The things correlations can be derived by mining the usage patterns. In the meantime, we craft the social information of users in terms of their connections with each other.

Things usage logs represent the interactions between users and things, and each log is created when a person interacts with a particular thing. The location $\mathcal{L}$ denotes the things' location information, where $\mathcal{S}$ is a set of time stamps, indicating that a user accesses a thing at a specific time in a specific location. Each usage event record is a triple of *ThingID, Timestamp, Location* as described in the following.

Definition 1 (*Things Usage Event*). Let $\mathcal{T} = \{t_1, \ldots, t_n\}$, $\mathcal{U} = \{u_1, \ldots, u_m\}$, $\mathcal{S} = \{s_1, \ldots, s_p\}$, and $\mathcal{L} = \{l_1, \ldots, l_q\}$ represent the set of things, users, timestamps, and locations, respectively. A usage event of a thing t, denoted by $h \in \mathcal{H} = \{h_1, \ldots, h_i\} = \{<t, u, s, l> | t \in \mathcal{T} \wedge u \in \mathcal{U} \wedge s \in \mathcal{S} \wedge l \in \mathcal{L}\}$, indicates that user u used thing t located in a specific location l at timestamp s.

Things recommendation in IoT can be formulated as predicting the dyadic relationships between people and things by leveraging informative relations from the thing-thing graph, the user-thing graph, and the user-user graph. In other words, our model merges three heterogeneous types of dyadic relationships, namely, *people to things* ($\{\exists y_{i,j}, \forall i \in \mathcal{P}, j \in \mathcal{T}\}$), *people to people* ($\{\exists s_{i,i'}, \forall i, i' \in \mathcal{P}\}$), and *things to things* ($\{\exists t_{i,i'}, \forall i, i' \in \mathcal{T}\}$), where $\mathcal{P}$ and $\mathcal{T}$ denote people and things, respectively. Figure 1 shows the graphical representation of the three relationships.

We aim at recommending certain things (e.g., Microwave 2 at lunch time in Figure 1) to users according to *predicted things usage value*, which is reflected by the usage frequency of things. In particular, a pair of user-thing instance $(i, j), \forall i \in \mathcal{T}, \forall j \in \mathcal{P}$ generates the interactive relationship, which can be abstracted as

$$\{(i, j) \rightarrow y_{ij}\} \text{ where } i \in \mathcal{T}, j \in \mathcal{P}, \tag{1}$$

which forms a matrix $Y \in \mathcal{Y}^{|\mathcal{I}||\mathcal{J}|}$. Our goal is to predict the missing entries $y_{\tilde{i}\tilde{j}}$ given testing pair $(\tilde{i}, \tilde{j})$. Things with bigger values of $y_{\tilde{i}\tilde{j}}$ will be recommended to a given user.

3. PROPOSED RECOMMENDATION FRAMEWORK

In this section, we introduce our unified framework, where the information from social relations and things correlations are coupled simultaneously through shared latent factors learned from three matrices: the *user-user relationship* matrix, the *thing-thing correlation* matrix, and the *user-thing usage* matrix. Our methodology fuses user relationships, things correlations, and user-thing interactions together, and incorporates three relationships: *user-user connections*, *thing-thing correlations*, and *user-thing interactions* (thing usage). We describe how to encode these three relationship matrices in our model from Section 3.1 to Section 3.2.

3.1. Decoding Users Relationship

We construct a directed weighted graph $\mathbf{G}_u = (V_u, E_v)$, whose vertex set V_u corresponds to users set $\{u_1, \ldots, u_m\}$; edges set E_v represents the relationships between users and the range of their associated weight are in [0, 1]. Bigger weights indicate stronger ties between users. The weight W_u indicates the user similarity influenced by the social links between users, reflecting the *homophily* (i.e., similar users may have similar interests). We use the cosine similarity to calculated $s_{ii'}$ as follows:

$$s_{ii'} = \frac{e^{\alpha cos(b(i), b(i'))}}{\sum_{k \in \Omega(i)} e^{\alpha cos(b(i), (b(k))}}, \tag{2}$$

where $cos(b(i), b(i')) = \frac{b(i) \cdot b(i')}{||b(i)||||b(i')||}$, $\Omega(i)$ is the set of the user i's connections (i.e., $j \in \Omega(i)$), $b(i)$ is the binary vector of things used by user i, $||\cdot||$ is the L-2 norm of vector $b(\cdot)$, and α is a parameter that reflects the preference for transitioning to a user who interacts with the same things.

After we obtain the users' relationship matrix from $\mathbf{G}_u$, we factorize users' relationship matrix to derive a high-quality, low-dimensional feature representation to user-based latent features vectors $u_i \in \mathbb{R}^{1\times m}$ and factor-based latent feature vectors $u'_{i'} \in \mathbb{R}^{1\times m}$ on analyzing the social network graph $\mathbf{G}_u$. The conditional probability of $s_{ii'}$ over the observed social network is determined by

$$\begin{aligned} & s_{ii'} \sim Pr\left(s_{ii'} | u_i^T u'_{i'}; \sigma_s\right) \\ & \text{where} \quad u_i \sim \mathcal{N}\left(0, \sigma_u^2\right), u'_{i'} \sim \mathcal{N}\left(0, \sigma_{u'}^2\right). \end{aligned} \tag{3}$$

Similar to the Web link adjacency, if a user i has lots of links to other users, the trust value of $s_{ii'}$ should be decreased. While if a user i is trusted by many other users, the trust value $s_{ii'}$ should be increased; since the user can be considered as local authority, $s_{ii'}$ should be adjusted as

$$s^*_{ii'} = \sqrt{\frac{d^-(i')}{d^+(i) + d^-(i')}} \times S_{ii'}, \tag{4}$$

where $d^+(i)$ represents the out-degree of node i, while $d^-(i')$ indicates the in-degree of i'. Equation (3) can be reformulated as

$$s^*_{ii'} \sim Pr\left(s^*_{ii'} | u^T u'; \sigma_s\right). \tag{5}$$

3.2. Decoding Things Correlations

Compared with users' social relations, things correlations are implicit and not straightforward to obtain, and there are some unique challenges in order to learn things correlations [Yao et al. 2013]. To fully utilize the rich information in things usage logs mentioned previously, we extract implicit things correlations by mining the underlying connections among things and their spatiotemporal patterns in this article.

Our study builds on the theory of *homophily* from the field of social networks, which implies that people with similar characteristics tend to form relationships [McPherson et al. 2001]. Then, the presence of relationships among people can be used to infer their similarities. Moreover, the stronger the tie, the higher the similarity. This inference is particularly useful when characteristics of people are not directly observable or incomplete. We advocate that the homophily principle applies to things as well, that is, things with strong interactive relationships tend to have strong correlations. We propose a unified hypergraph to represent the various entities and heterogeneous relations in collection of things usage events in terms of temporal and spatial dimensions; hypergraph is a generalization of the ordinary graph, where the edges, also known as *hyperedges*, are arbitrary nonempty subsets of the vertex set [Zhou et al. 2006; Agarwal et al. 2006]. With hypergraph modeling, the vertices represent various types of entities in user-thing interactions (e.g., people, things, temporal and spatial information, etc.), and the hyperedges represent heterogeneous relations among entities via connecting arbitrary subsets of vertices. In this way, we not only incorporate heterogeneous relations, for example, the similarity between locations, but also utilize the spatiotemporal historical interactions on things collections. On top of this unified hypergraph, we develop a ranking-based algorithm to learn the most related things according to a query by propagating the information of target thing through the structure of this constructed hypergraph. We take an example to show processing of things

Table I. Samples of One User's Things Usage Log

ThingID	Timestamp	Location
1	20101108020307	Office-1
5	20101109073226	Sec4-KitchenHome
5	20101109074235	Sec4-KitchenHome
5	20101109075139	Sec4-KitchenHome
4	20101109091202	Sec2-LivingRoom1
4	20101109091408	Sec2-LivingRoom1
2	20101109171643	Sec4-KitchenHome
...	...	...

Table II. Processed Samples of One User's Things Usage Log

ThingID	Timestamp	Location
t1	s1	p2
t5	s4	p4
t5	s4	p4
t5	s4	p4
t4	s3	p1
t4	s3	p1
t2	s2	p4
...	...	...

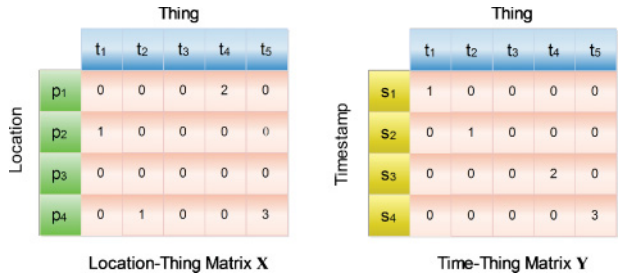


Fig. 2. Spatiotemporal matrices extracted from samples shown in Table I.

usage logs as follows. Table I shows some raw samples of things usage events in terms of time and location dimensions, and Table II shows the same samples after processing where things, locations, and timestamps are represented using unique IDs. Note that the second, the third, and the fourth timestamps fall into a same time period, and they are therefore given the same timestamp ID. Figure 2 reflects the location-thing and time-thing matrices extracted from logs.

Given things usage logs in terms of temporal and spatial factors, thing-thing correlation derivation can be formally formulated as Problem 1.

PROBLEM 1 (THINGS CORRELATION DERIVATION). *Given each thing denoted as a vector y_i whose initial entry corresponding to the target thing is 1, 0 otherwise, its relations of the rest of other things related to the given thing can be quantified by learning a ranking function.*

To solve this problem, there are two subproblems that need to be solved sequentially, defined as subproblem 1 and subproblem 2, respectively.

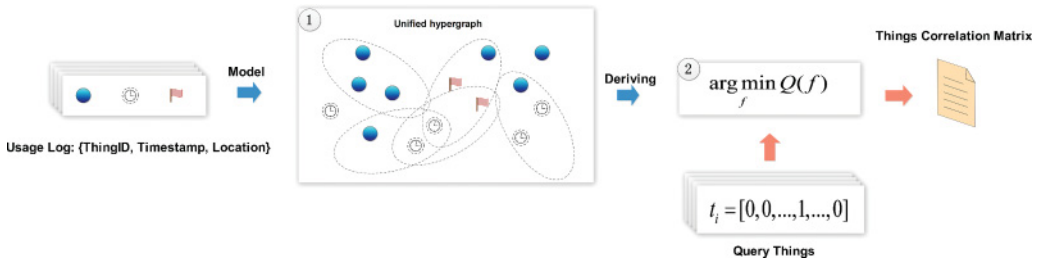


Fig. 3. Overview of the Proposed Things Correlation Derivation: stage ① is to represent heterogeneous entities and relations (e.g., things, locations, and timestamps) in things usage events to a unified hypergraph; stage ② is to develop a ranking algorithm to search related things (the blue arrows show the training stage while the orange arrows show the testing stage).

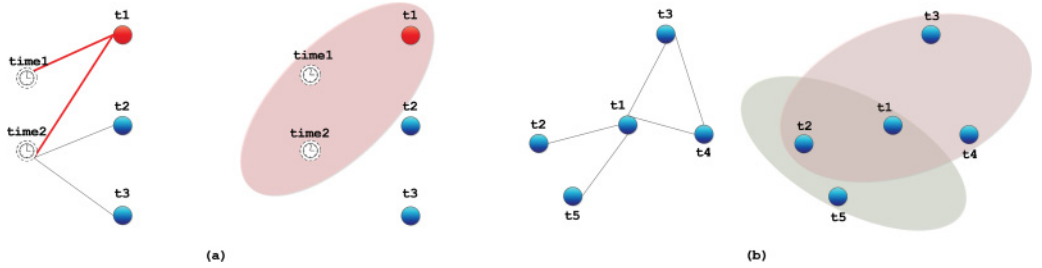


Fig. 4. Illustrative example of hypergraph versus conventional graph on modeling things usage events in terms of unified relations and high-order relations: (a) the conventional graph (left) in which a pair of user and thing is connected if the user accessed this thing. This graph cannot clearly reflect whether the graph measures pairwise relations, for example, $<time_1, t_1>$ and $<time_2, t_1>$, and the hypergraph (right), where each hyperedge can contain any number of arbitrary nodes to exhibit more complex relations, for example, hyperedges can be written as $<time_1, time_2, t_1>$ (ellipse shadow), from which we can clearly know thing t_1 is accessed at $<time_1, time_2>$; (b) the conventional graph (left) in which two things can be connected if they are accessed at same time interval, but it cannot tell whether this thing is accessed at other time frames, and the hypergraph (right) can reasonably represent this complex relation.

SUBPROBLEM 1 (MODELING). *Given a collection of things usage events $\mathcal{H}$, construct a hypergraph-based model HG representing the heterogeneous entities and their complex relations across and within the entities.*

SUBPROBLEM 2 (INFERENCE). *Given the constructed hypergraph HG induced from things usage events collection $\mathcal{H}$, learn a ranking function that can produce a list of related things for each target thing.*

These two subproblems have successive relations. Ideally, a solution should be *robust* in capturing and reflecting the hidden pattern in usage events, and also be *efficient* in ranking the candidates.

Our proposed solution (see Figure 3) can be decomposed into two stages: (i) *descriptive stage* and (ii) *predictive stage*. The first stage aims at embracing the multiple relations in things usage events, and we particularly propose a hypergraph to model the heterogeneous entities and relations of things usage events (corresponding to Subproblem 1). In the second stage, we infer a list of most related things given a querying thing by developing a ranking algorithm on top of the descriptive model in stage 1 (corresponding to Subproblem 2).

3.2.1. Hypergraph Preliminary. As a generalization of the conventional graphs, a hypergraph can naturally address these mentioned challenges (as demonstrated in Figure 4).

The fundamental idea is to explore the underlying similarity relationships among vertices via constructing a hypergraph with various hyperedges to capture the high-order and complex similarities. Hypergraph has been extensively explored in recommendation [Bu et al. 2010] and multilabel classification [Sun et al. 2008].

Let $HG(\mathcal{V}, \mathcal{E})$ be a hypergraph with the vertex set $\mathcal{V} = \mathcal{T} \cup \mathcal{L} \cup \mathcal{S}$ and the set of hyperedges $\mathcal{E}$. A hyperedge e is a nonempty subset of $\mathcal{V}$ where $\cup_{e \in \mathcal{E}} = \mathcal{V}$. Let $HG(\mathcal{V}, \mathcal{E}, \mathcal{W})$ be a weighted hypergraph where $\mathcal{W} : \mathcal{E} \to \mathbb{R}^+$ is the hyperedge weight. The hypergraph is said to be connected when there is a path between each pair of vertices. A path is a connected sequence of vertices over hyperedges $\{v_1, e_1, v_2, e_2, \ldots, e_{k-1}, v_k\}$ where $\{v_i, v_{i+1}\} \subseteq e_i$. A hyperedge e is said to be incident with v when $v \in e$. A hypergraph has an incidence matrix $\mathcal{H} \in \mathbb{R}^{|\mathcal{V}| \times |\mathcal{E}|}$ and each entry $h(v, e)$ is defined as follows [Bu et al. 2010; Zhou et al. 2006]:

$$h(v, e) = \begin{cases} 1 & \text{if } v \in e \\ 0 & \text{if } v \notin e. \end{cases} \tag{6}$$

So, the degree $d(v)$ of vertex v and the degree $\delta(e)$ of hypergraph degree e are defined as

$$\begin{aligned} d(v) &= \sum_{e \in \mathcal{E}} \mathcal{W}(e) h(v, e), \\ \delta(e) &= \sum_{v \in \mathcal{V}} h(v, e) = |e|, \end{aligned} \tag{7}$$

where $\mathcal{W}(e)$ is the weight of the hyperedge e, and $\delta|e|$ is the number of vertices in e. We use $\mathcal{D}_e$ and $\mathcal{D}_v$ to denote the diagonal matrices of the degrees of hyperedges and vertices, respectively.

3.2.2. Subproblem 1: Modeling. We first introduce some intuitions behind using hypergraph to model things usage events, and then describe how to realize the modeling task defined in subproblem 1.

As mentioned before, things usage event is a four-elemental tuple containing *object, user, timestamp, location*, which means there are four types of entities and multiple relationships across these entities. We summarize three major challenges in order to exploit and model the rich information:

—*How to model heterogeneous relations with minimal information loss.* The various entities and relations make it difficult to develop a *unified* framework taking into account all objects and relations. For example, there are three types of contextual attributes in things usage events. To illustrate their relations with objects, we generally need to construct three separate graphs, for example, location-thing graph and time-thing graph. However, dividing multiple relations into separated graphs can cause nontrivial information loss.

—*How to robustly model high-order relations.* There exist some relations that are more complex, and it will cause information loss if simply being modeled using a low-order modeling approach, for example, using pairwise similarity based graphs. Plus, a pairwise relation between two nodes is only based on their own characteristics. If one of them is corrupt, their relation will be broken. As a result, the relations between pairwise nodes are not stable.

—*How to reveal latent connections.* The complex dependencies among a variety of entities make it hard to detect a latent relationship between entities. For example, although the usage frequency of thing $<t_1, t_3>$ being used together falls below some threshold, they might actually be meaningfully related. Let us say things $<t_3, t_2>$ are used together frequently, and $<t_1, t_2>$ are also used frequently together. The presence

Table III. Hyperedges in Things Usage Events

External Relations across Entities		
Index	Notation	Hyperedges
1	$\mathcal{E}^{\mathcal{TTI}}$	Thing-Thing-Time
2	$\mathcal{E}^{\mathcal{TTL}}$	Thing-Thing-Location
3	$\mathcal{E}^{\mathcal{LLT}}$	Location-Location-Thing
4	$\mathcal{E}^{\mathcal{ITT}}$	Time-Time-Thing
Internal Relations within Entities		
Index	Notation	Hyperedges
5	$\mathcal{E}^{\mathcal{L}^p}$	p-nearest locations

of t_2 actually provides some possible latent connection for $<t_1, t_3>$, which needs to be uncovered.

Conventional graphs can overcome these obstacles partially. However, it is difficult to fully capture the heterogeneous and complex relations in things usage events by using conventional graphs. For instance, the underlying cross-entity relationships in things usage events usually have richer structures than conventional graphs can contain. Figure 4(a) shows a simple example to compare the scenarios of using conventional graph (left) and of using hypergraph to model temporal relations of things (right). In this example, a thing t_1 is accessed at $time_1$ and $time_2$. The conventional graph obviously cannot capture this connection; for example, to show $time_1$ and $time_2$ relations with t_1, respectively, it needs two edges $<time_1, t_1>$ and $<time_2, t_1>$. From this representation, we cannot derive relations whether t_1 and t_2 are accessed in the same time interval. However, the hypergraph can tackle this problem naturally due to the fact that its edges can contain any number of arbitrary nodes. The hyperedge $<time_1, time_2, t_1>$ clearly demonstrates the relations among the entities. Figure 4(b) shows one thing is highly possible to be accessed at multiple specific time frames. The conventional graph (left) describing binary correlations cannot capture this high-order relation, however, hypergraph (right) can accomplish this complex relation naturally.

A hyperedge in our things usage hypergraph can be a set of vertices with either the same type or different types. The former kind of hyperedge captures the relations among the same type of objects, while the latter captures the relations across different types of objects. Things usage events consist of four types of entities. Let $\mathcal{T}$ denote the set of things, $\mathcal{I}$ denote the set of timestamps, and $\mathcal{L}$ denote the set of locations. In our data model, we formalize a hypergraph HG that contains six different implicit and complex relations with different entities, namely, *external* relations across entities. Another two high-order relations are constructed within two types of entities including users and locations, which we call *internal relations*. Table III summarizes all the hyperedges modeled by our unified hypergraph. We briefly introduce them in the following.

External Relations. There are four external relations in our model:

—$\mathcal{E}^{\mathcal{TTI}}$. This represents the scenario that two things are used at a same time period. The weight $w(e^{t_i t_j i_k})$ for this relation is set to be the frequency that both things t_i and t_j are used in same time period i_k. The calculation is the same as Equation (9),

$$w(e^{t_i t_j i_k}) = |\{(t_i, t_j, i_k)\}| t_i \in \mathcal{T}, t_j \in \mathcal{T}, i_k \in \mathcal{L}| \tag{8}$$

and it can be normalized as

$$w(e^{t_i t_j i_k}) = \frac{w(e^{t_i t_j i_k})}{\sqrt{\sum_{l=1}^{|\mathcal{T}|} w(e^{t_l t_j i_k})}\sqrt{\sum_{m=1}^{|\mathcal{T}|} w(e^{t_i t_m i_k})}}. \tag{9}$$

Table IV. The Incident Matrix $\mathcal{H}$ of Our Proposed Hypergraph

	$\mathcal{E}^{\mathcal{TTI}}$	$\mathcal{E}^{\mathcal{TTL}}$	$\mathcal{E}^{\mathcal{LLT}}$	$\mathcal{E}^{\mathcal{ITT}}$	$\mathcal{E}^{\mathcal{L}^p}$
$\mathcal{T}$	$\mathcal{TE}^{\mathcal{TTI}}$	$\mathcal{TE}^{\mathcal{TTL}}$	$\mathcal{TE}^{\mathcal{LLT}}$	$\mathcal{TE}^{\mathcal{ITT}}$	0
$\mathcal{I}$	$\mathcal{IE}^{\mathcal{TTI}}$	0	0	$\mathcal{IE}^{\mathcal{ITT}}$	0
$\mathcal{L}$	0	$\mathcal{LE}^{\mathcal{TTL}}$	0	$\mathcal{LE}^{\mathcal{LLT}}$	$\mathcal{LE}^{\mathcal{L}^p}$

—$\mathcal{E}^{\mathcal{TTL}}$. This represents that two things are used at a same location. The weight $w(e^{t_i t_j l_k})$ for this relation is set to be the frequency that both things t_i and t_j are used in the same location l_k. The calculation is similar to Equation (9).
—$\mathcal{E}^{\mathcal{ITT}}$. In this relation, if one thing is used at the same timestamp, we assign the weight as 1.
—$\mathcal{E}^{\mathcal{LLT}}$. In this relation, if one thing is used in the same location, we assign the weight as 1.

Internal Relations.

—$\mathcal{E}^{\mathcal{L}^p}$. In this relation, we consider the similarity between different locations. In the hypergraph, a hyperedge of this type is each location and its top p similar locations. For each location l_i, its weight of hyperedge is calculated as

$$w(e^{l_i^p}) = \frac{1}{p}\sum_{j=1}^{p} sim(l_i, l_j), \tag{10}$$

where $sim(l_i, l_j)$ is the similarity between two locations. Given two locations, we measure their similarity using the Jaccard coefficient between the sets of things used at each location:

$$sim(l_i, l_j) = \frac{|\Gamma_i^o \cap \Gamma_j^o|}{|\Gamma_i^o \cup \Gamma_j^o|}, \tag{11}$$

where Γ_i^o and Γ_j^o denote the set of used things at location l_i and location l_j, respectively.

Based on the hypergraph model introduced previously, we can derive the vertex-hyperedge incidence matrix $\mathcal{H}$ (as shown in Table IV) and also the weight matrix $\mathcal{W}$. The size of both matrices depends on the cardinality of different element sets involved in the matrices, and they are all sparse matrices.

3.2.3. Subproblem 2: Inference. To infer the related things for a given querying thing, we adopt a spectral clustering based semisupervised learning framework [Zhu 2006]. The goal is, given each thing i, to estimate a ranking function $\mathbf{f}$ that allocates other things with different relatedness scores $f(i)$ over hypergraph HG, indicating their relevance with querying thing $i \in \mathcal{V}$. Let $t_i \in \mathbf{t}$ be the query vector for thing i. The learning process can be formulated under a regularization framework as follows:

$$Q(\mathbf{f}) = E(\mathbf{f}) + \mu\ell(\mathbf{f}, \mathbf{t}), \tag{12}$$

where $E(\mathbf{f})$ is the quadratic energy function, which smooths the vertices nearby according to Markov assumption; in other words, the nearby vertices on HG should have similar relatedness scores, and $\ell(\mathbf{f}, \mathbf{t})$ is the loss function measuring the difference between predicted relatedness score and true scores. μ is the regularizer. The equation

ALGORITHM 1: *Hypergraph-Based Things Correlation Learning*

Input: A sequence of things usage events $\mathcal{H}$ and things set $\mathbf{t} = \{t_1, \ldots, t_n\}$
Output: Relation matrix of things $\mathcal{T}$

1 Constructing hypergraph HG as described in Section 3.2.2;
2 Computing incident matrix $\mathcal{H}$;
3 Computing weight matrix $\mathcal{H}$;
4 Computing vertice degree matrix $\mathcal{D}_v$;
5 Computing hyperedge degree matrix $\mathcal{D}_e$;
6 Computing $\Theta = \mathcal{D}_v^{-1/2}\mathcal{H}\mathcal{W}\mathcal{D}_e^{-1}\mathcal{H}^T\mathcal{D}_v^{-1/2}$;
7 Initializing $\mathcal{T} = \varnothing$;
8 **for** *i=1:n* **do**
9 Initializing relatedness score vector $\mathbf{t}_i$ as: the i-th element is $t_i^i = 1$, other elements are equal to 0;
10 Let $\mathbf{f}^0 = t_i$;
11 **while** $\delta < \xi$ **do**
12 $\mathbf{f}^{j+1} = \alpha\Theta\mathbf{f}^j + (1-\alpha)t_i$;
13 **for** *k = 1:n* **do**
14 $\delta \leftarrow \max|\mathbf{f}_k^{j+1} - \mathbf{f}_k^j|$;
15 **end**
16 **end**
17 Output $\{t_1, \ldots t_n\}$ sorted by relatedness score with $\mathbf{t}_i$;
18 $\mathcal{T} = \mathcal{T} \cup \mathbf{t}_i$;
19 **end**

can be expanded as

$$Q(\mathbf{f}) = \frac{1}{2}\sum_{i,j=1}^{|\mathcal{V}|}\sum_{e\in\mathcal{E}}\frac{1}{\delta(e)}\sum_{(v_i,v_j)\subset e}\mathcal{W}(e)\left|\frac{\mathbf{f}_i}{\sqrt{d(v_i)}} - \frac{\mathbf{f}_j}{d(v_j)}\right|^2 + \mu\sum_{i=1}^{|V|}||\mathbf{f}_i - t_i||^2. \tag{13}$$

The optimal relatedness scores can be learned recursively by solving the optimization problem in a closed form [Zhou et al. 2006]:

$$\begin{aligned}&\mathbf{f}^* = \text{argmin}_{\mathbf{f}}Q(\mathbf{f}) = (1-\alpha)(1-\alpha\Theta)^{-1}\mathbf{t}\\ &\quad\text{where}\\ &\quad\alpha = 1/1+\mu \text{ and } \Theta = \mathcal{D}_v^{-1/2}\mathcal{H}\mathcal{W}\mathcal{D}_e^{-1}\mathcal{H}^T\mathcal{D}_v^{-1/2}.\end{aligned} \tag{14}$$

Equation (14) can be reformulated as

$$\mathbf{f}^* = \alpha\Theta\mathbf{f}^* + (1-\alpha)\mathbf{t}. \tag{15}$$

Our things correlation approach is implemented in a two-stage process. We first integrate both interactive relationships formulated across things, timestamps, and locations, and pairwise relationships formulated between any homogeneous entities from the things usage events in a unified hypergraph. Then, given the hypergraph representation of things usage events, we treat correlation derivations as a ranking problem over the hypergraph, where each thing is treated as a query $\mathbf{t}_i$ (line 9 in Algorithm 1). Our task is to rank other nodes on the hypergraph according to their related score to the query node (from line 8 in Algorithm 1). Following this process, we can derive pairwise relations of things. The detailed algorithm of things derivation is summarized in Algorithm 1.

3.3. Decoding User-Thing Interactions

User-thing interactions y_{ij} are embodied by the usage frequency of thing i by user j in a certain time span. We can map the usage frequency to interval [0, 1] by using function $f(x) = (x - y_{min})/(y_{max} - y_{min})$ without loss of generality, where y_{max} and y_{min} are the maximum and minimum usage values, respectively. The dyadic relationship between a user and a thing does not only depend on their latent factor $U^T V$, whose vulnerability is that it makes use of past interactions and cannot handle brand new things well, that is, cold-start problem. To tackle this issue, we use the explicit features directly by profiling users observable features $x_i \in \mathbb{R}^c$ (i.e., age, gender, location, etc.) and things observable features $z_j \in \mathbb{R}^d$ (i.e., textual description of things functionalities, things contextual information, etc.) [Yang et al. 2011b]. Here c and d are the dimensionality of users observable features and things observable features, respectively. The dyadic relationship (thing usage value) depends on not only the inner product of latent factors of users and things, but also their observable features. Things usage value y_{ij} can be defined as the following conditional probability:

$$y_{ij} \sim Pr\left(y_{ij} | u_i, v_j, x_i, z_j, \sigma_y^2\right). \tag{16}$$

We adopt the bilinear product to specify the similarity between user observable features and thing observable features [Chu and Park 2009]. The pairwise similarity between x_i and z_j can be denoted as

$$r_{ij} = \mathbf{w}^T \left(x_i \otimes z_j\right), \tag{17}$$

where $\mathbf{w}$ is a column vector of entries $\{w_{mn}\}$, and $x_i \otimes z_j$ denotes the Kronecker product of x_i and z_j. Equation (17) can be rewritten as

$$r_{ij} = x_i^T W z_j, \tag{18}$$

where matrix $W \in \mathbb{R}^{m \times n}$ is a weight coefficients capturing pairwise associations between user i's explicit feature vector and thing j's explicit feature vector including key state indicator or descriptive information of things. The explicit feature vector x_i of user i reflects demographic information of users like gender, identity, etc.

The explicit feature vector z_j of things includes static features and dynamic features. The static features contain textual descriptors of things, and property information. Since the dynamic feature consists of things availability that changes over time, we adopt a *particle filtering* based tracking method to obtain the latest things' availability [Yao and Sheng 2011], which indicates things availability temporally.

So the thing usage value depends on both the inner product of user and thing latent factors and the bilinear product of user observable features and thing observable features. Equation (16) can be reformulated as

$$y_{ij} \sim Pr\left(y_{ij} | u_i^T v_j + r_{ij}, \sigma_y^2\right). \tag{19}$$

3.4. The Fusion Model

Given observable data for $\mathbb{O} = \{\mathbb{O}_y, \mathbb{O}_s, \mathbb{O}_t\}$, our framework can be denoted as an optimization problem of minimizing the negative logarithm of $Pr(\mathbb{O}|\Sigma)Pr(\Sigma)$ via

$$\begin{aligned} \min_{\Sigma} L(\Sigma) = \min \lambda_y \sum_{ij \in \mathbb{O}_y} \ell\left(y_{ij}, u_i^T v_j + r_{ij}\right) + \lambda_s \sum_{ii' \in \mathbb{O}_s} \ell\left(s_{ii'}^*, u_i^T u_{i'}'\right) + \lambda_t \sum_{jj' \in \mathbb{O}_t} \ell\left(t_{jj'}, v_j^T v_{j'}'\right) \\ + \lambda_{\mathbf{w}} ||\mathbf{w}||^2 + \lambda_U ||U||^2 + \lambda_V ||V||^2, \end{aligned} \tag{20}$$

where $\ell\cdot$ is a loss function (we adopt the most widely used ℓ_2 loss). A gradient descent process can be implemented to solve the parameters. Given a training dataset $\{y_{ij}\}$, the

objective function in Equation (20) can be found by performing gradient descent in u_i, v_j, and w_{mn}.

$$\begin{aligned} u_i &\rightarrow u_i - \delta \times \left(\frac{\partial \ell}{\partial u_i}(y_{ij}, u^T v + r_{ij}) v_j + u_i^T u_i \right), \\ v_j &\rightarrow v_j - \delta \times \left(\frac{\partial \ell}{\partial v_j}(y_{ij}, u^T v + r_{ij}) u_i + v_j^T v_j \right), \\ w &\rightarrow w - \delta \times \left(\frac{\partial \ell}{\partial w}(y_{ij}, u^T v + r_{ij}) u_i v_j^T + w^T w \right), \end{aligned} \tag{21}$$

where δ is the learning rate. After we obtain the optimal parameters Σ^*, we can use them to predict the given testing data $\{\tilde{i}, \tilde{j}, y_{\tilde{i},\tilde{j}}\}$:

$$y_{\tilde{i}\tilde{j}} = u_{\tilde{i}}^T v_{\tilde{j}} + \sum_{mn}^{cd} x_{\tilde{i}n} z_{\tilde{j}m} w_{mn}^*, \tag{22}$$

where c and d are the dimensionality of users' explicit features $\mathbf{x}$ and things explicit features $\mathbf{z}$.

Our proposed fusion model couples three types of relations: user social relations, thing correlations, and user-thing relations. The model is constrained by three types of information: similar user regularization, similar thing correlation, and similar cohesion between users and things. We use the aforementioned information to help better shape the user and thing latent representative spaces to generate more accurate recommendation results on things of interest.

4. EVALUATIONS

We conducted several experiments to evaluate the performance of our proposed approach. The first experiment is to compare our approach with other state-of-the-art methods. The second experiment is to demonstrate the impact of social networks and things correlations on recommendation, respectively. The third experiment demonstrates the impact of different methods of things correlation derivation. We also study the impact of different time granularity decomposition in the fourth experiment.

4.1. Experimental Settings

We set up a testbed that consists of several different places (e.g., bedroom, bathroom, garage, kitchen), where approximately 127 physical things (e.g., couch, laptop, microwave oven, fridge) are monitored by attaching RFID tags and sensors. This task greatly benefits from our extensive experience in a large RFID research project [Wu et al. 2012]. We have developed a research prototype that provides an environment where users can check and control things real time via a Web interface.[1] Figure 5(a) shows some RFID devices and sensors used in the implementation and Figure 5(b) shows part of the kitchen setting in our testbed. In our implementation, things are exposed on the Web using RESTful Web services, which can be discovered and accessed from a Web-based interface.

Figure 6 shows the architecture of our testbed. The system features a layered architecture, and is developed using the Microsoft .NET framework and SQL Server 2012. Physical things and their related data and events are mapped to corresponding virtual resources, which can be aggregated and visualized via a range of software components. The system provides two ways to identify physical objects and connect them to the

[1]https://www.youtube.com/watch?v=t4DHt0vUulY.

Fig. 5. The settings of system: (a) some sensors and RFID devices, and (b) part of the kitchen setting (e.g., sensor-enabled microwave oven and toaster).

Web. The first one is to use RFID technology, where the physical objects are attached with RFID tags and interrogated by RFID readers. The second one is to combine sensors with objects to transfer the raw data to the network. The raw data captured by readers and sensors will be further processed. The data access layer (i) manages RFID tags and sensors associated with physical things, (ii) collects raw RFID and sensor data and processes them, and (iii) provides a universal API for higher level programs to retrieve the status of things. Due to inherent characteristics of RFID and sensor data (e.g., volatile) [Sheng et al. 2008], this layer contains several software components for filtering and cleaning the collected raw data, and adapting such data for high-level applications. The advantage of the data access layer is to allow the system to provide data synchronously. This is important since some devices work with more than one sensor and the sensor readings may come asynchronously. This layer works in a scalable plug-and-play fashion, where new sensors can be easily plugged in and the existing sensors can be removed.

The Virtual Things module maps a collection of classes (also called virtual things) to their corresponding physical things. Each virtual thing communicates with the sensor hive, collects the information, and interprets the current status of the corresponding physical device. The Event Management layer focuses on event processing that automatically extracts and aggregates things usage events based on the data feeds from the virtual things layer in a pipelined fashion. The pipeline consists of three main phases: the event detection, the contextual information retrieval, and the event aggregation.

In our implementation, there are two ways to detect usage events of things: *sensor-based* for detecting state changes and *RFID-based* for detecting mobility [Yao et al. 2013]. In the sensor-based detection, the usage of an object is reflected by changes of the object's status, for example, a microwave oven moved from an idle to an in-use state. In the RFID-based detection, the movement of an object indicates that the object is being used. For example, if a coffee mug is moving, it is likely that the mug is being used. In this situation, we adopt a generic method based on comparing descriptive statistics of the Received Signal Strength Indication (RSSI) values in consecutive sliding windows [Parlak et al. 2011]. The statistics obtained from two consecutive windows are expected to differ significantly when an object is moved.

The Contextual Information Retriever fetches contextual information contained in things usage events. In our current implementation, we focus on three types of contextual information: *identity* (user), *temporal* (timestamp), and *spatial* (location) information [Yao and Sheng 2012; Yao et al. 2013]. For the spatial information, we consider two situations. For *static* objects (e.g., refrigerator), the spatial information is a prior knowledge. For *mobile* objects (e.g., RFID-tagged coffee mug), we provide coarse-grain

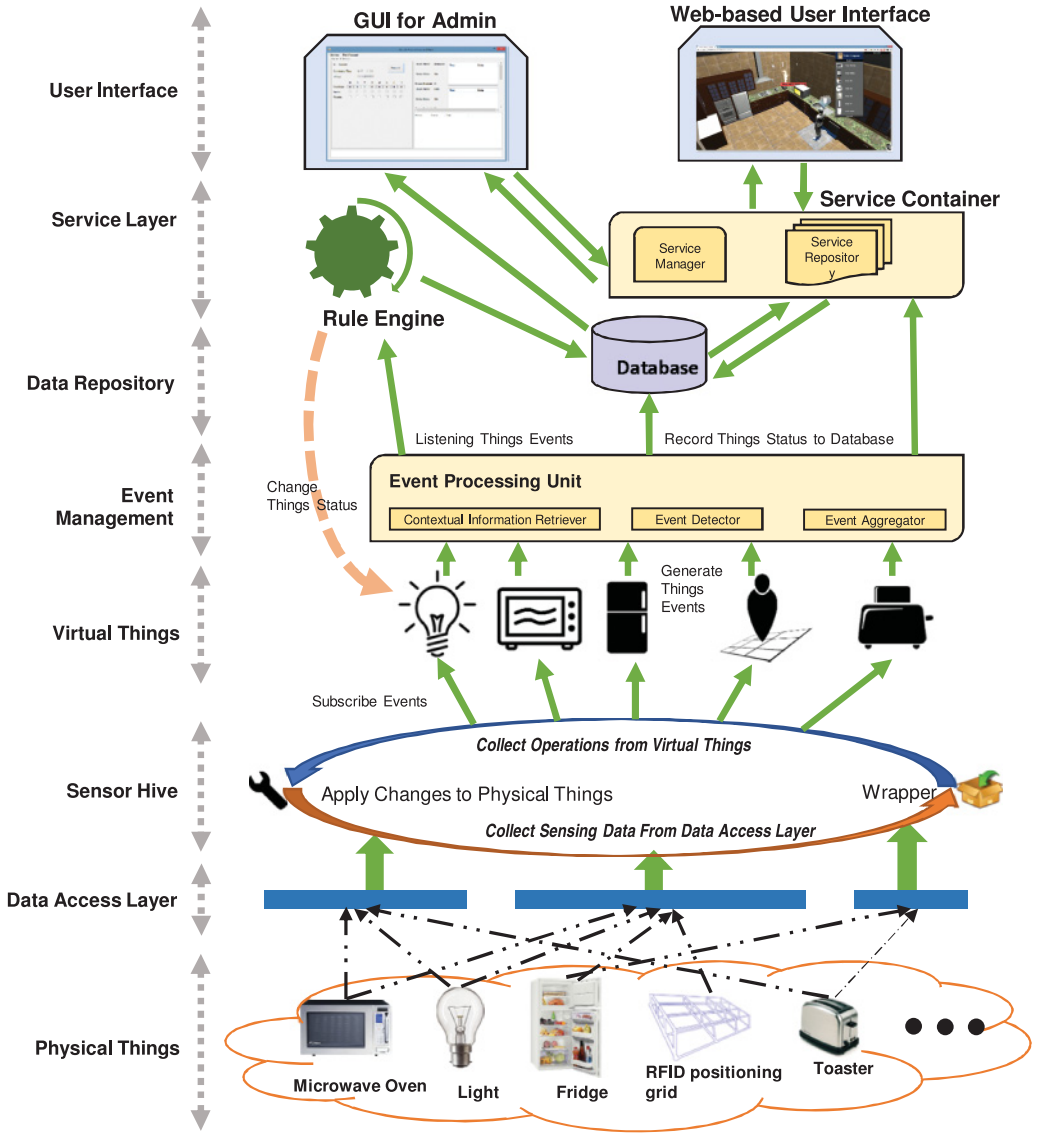


Fig. 6. The system architecture.

and fine-grain methods for localization. The coarse-grain method uses the RSSI signal received from a tagged object to approximate its proximity to an RFID antenna. Each zone is covered by a mutually exclusive set of RFID antennas. The zone scanned by an antenna with the maximum RSSI signal is regarded as the object's location. The fine-grain method compares the signal descriptors from an object at an unknown location to a previously constructed radio map or fingerprint. We use the Weighted k Nearest Neighbors algorithm (w-kNN) to find the most similar fingerprints and compute a weighted average of their 2D positions to estimate the unknown tag location [Yao et al. 2013].

Table V. Statistics of the Dataset

No.	Category	# Things	# Scale
1	Entertainment	44	68–1132
2	Office	26	511–1790
3	Cooking	42	54–2442
4	Transportation	13	31–870
5	Medicine/Medical	10	22–89
6	House Appliances	44	46–2863

Finally, the Service Layer consists of a rule engine and a service container. The service container converts events and data into corresponding services. In particular, the service repository stores the descriptions of services (in the form of RESTful APIs) for things, where applications can easily access the data associated with a particular thing stored in the database (e.g., usage history of a device), as well as manipulate the actuators (e.g., turning on or off a light). The APIs are represented using JavaScript Object Notation (JSON), which is developed from the JavaScript for representing simple data structures and associated objects. The service manager is responsible for abstracting services from the lower level, representing them as services, and storing them into the service repository.

Ten volunteers participated in the data collection by interacting with RFID-tagged things for a period of 4 months, generating 20,179 records on the interactions of tagged things. The dataset collected from our testbed is still small. To make the dataset big enough for experimental studies, we augmented our dataset with Washington State University's CASAS datasets,[2] which were also collected from a smart home environment. More specifically, we used dataset1 [Cook and Schmitter-Edgecombe 2009] and dataset2 [Singla et al. 2010]. From the datasets, we identified 52 more things (e.g., bowel, door, coffee table, and water tap) and deduced things usage events. For the location information, we referred to the sensor layout of each dataset for grouping sensors into corresponding locations. For example, L-11 to L-15 are located in the bathroom, so we mapped the location of things usage events related to L-11 to L-15 to the bathroom. Each activity was transformed into things usage events, to be used in our experiments. For example, consider the activity of "reading a magazine in couch," we converted it into two events, $<magazine, person1, timestamp, livingroom>$ and $<couch, person1, timestamp, livingroom>$. The detailed data preparation process can be found in Yao et al. [2014a] and Table V is a summary of the statistics of the dataset. It should be noted that "scale" in the table refers to the usage frequency between the least frequent used things and the most frequent used things. For example, for Cooking, one of the least used things is a blender (54 times), while one of the most used things is a fridge (2,442, the door open/close times of the fridge).

We adopted Mean Absolute Error (MAE) to measure the accuracy of our approach. MAE calculates the average of absolute difference between predicted usage values and actual values as the following:

$$MAE = \frac{\sum_{ij} |y_{ij} - \tilde{y}_{ij}|}{n}, \tag{23}$$

where y_{ij} is the actual thing's usage values between user i and thing j, $\tilde{y}_{ij}$ is the predicted thing's usage value, and n is the number of the predicted thing's usage values. The lower the MAE, the better the performance. Due to the size of our dataset, in our experiments, we calculated MAE based on the overall MAE on prediction things usage value across all categories.

[2]http://ailab.wsu.edu/casas/datasets/.

Table VI. MAE Comparison with Other Approaches on All Categories

Training Data	10%		20%		50%	
# of Factors	5	10	5	10	5	10
NMF	0.8803	0.8734	0.8602	0.8535	0.8517	0.8477
PMF	0.8767	0.8715	0.8511	0.8208	0.7917	0.7749
SVD++	0.8734	0.8514	0.8404	0.8268	0.7673	0.7426
SoRec	0.8287	0.7916	0.7956	0.7772	0.7507	0.7326
FST	0.7903	0.7746	0.7712	0.7648	0.7231	0.7106
FST/S	0.7894	0.7867	0.7747	0.7710	0.7273	0.7217
FST/T	0.7927	0.7893	0.7782	0.7739	0.7311	0.7304

4.2. Results

4.2.1. Performance Comparison. This experiment compares the prediction accuracy of our proposed approach based on Fusing Social networks and Things correlations (FST) with some state-of-the art approaches based on probabilistic factor analysis, namely, Probabilistic Matrix Factorization (PMF) [Salakhutdinov and Mnih 2007], SoRec [Ma et al. 2008], and SVD++ [Koren 2008].

—NMF applies nonnegative matrix factorization on user-location matrix to predict the possibility of thing usage. The user-thing interaction matrix can be decomposed into two lower dimension matrices in this method:

$$\min_{\mathbf{U},\mathbf{V}} \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{n} \left(y_{ij} - u_i^T v_j\right)^2. \tag{24}$$

—PMF is briefly defined as

$$y_{ij} \sim Pr\left(u_i^T v_j, \sigma_y\right), \tag{25}$$

where u_i and v_j are the low-dimensional factors learned from user-item interactions.

—SoRec integrates users' social network structure and the user-item interaction matrix. The integration is based on the probabilistic factor analysis through the shared user latent feature space u_i, by learning the low-rank user latent feature space u_i and $u'_{i'}$ on social network, and the item latent feature space v_j on user-item interaction matrix. It can be defined as

$$\begin{aligned} s_{ii'} &\sim p\left(s_{ii'} | u_i^T u'_{i'}, \sigma_r\right), \\ y_{ij} &\sim p\left(y_{ij} | u_i^T v_j, \sigma_y\right). \end{aligned} \tag{26}$$

—SVD++ combines the neighborhood models and latent factor models together.

$$y_{ij} \sim p\left(y_{ij} | \hat{u}_i^T v_j, \sigma_y\right), \tag{27}$$

where $\hat{u}_i = \frac{\sum_{i' \in \mathcal{N}_i} \omega_{ii'} u'_{i'}}{\sum_{i' \in \mathcal{N}_i} \omega_{ii'}}$, and $\mathcal{N}_i$ refers to neighbor of user i.

This experiment evaluated our approach, in particular its capability in handling the cold-start problem, an important issue in recommender systems that refers to providing accurate prediction when some users only use few things or even have no usage historical records at all. In order to verify the capability of our approach on predicting usage value of things that have not been used, we randomly selected and marked off $p\%$ of data (p= 10, 20, and 50) from our dataset as training data and different number of latent factors (5 and 10) to test all the methods. The experimental results are shown in Table VI.

From the table, it is clear that our approach outperforms other methods on different training ratios and different number of factors, especially when the training data is small, that is, when the training ratio is 10%. The reason lies in PMF is a pure probabilistic factor model. Relying heavily on user-thing usage matrix, it cannot deal with the circumstance where little interactions information is available. SoRec works better than PMF and SVD++ because of its aggregation of user-user internal information (social links). Our approach not only incorporates users and things internal information, but also defines the explicit features (i.e., content) for users (e.g., users profile) and things (e.g., description of things functionalities), which makes our approach perform better when there is a cold-start problem. The experimental result further demonstrates the effectiveness on improving the recommendation accuracy by incorporating things correlations.

4.2.2. Impact of Things Correlations and Social Influence. This experiment evaluated the impact of users' relations and things correlations to our model. To do so, we excluded the things correlations information from our model, denoted as FST/T, and the social relations from the model (FST/S), respectively. FST/T is defined as

$$\begin{aligned} y_{ij} &\sim Pr\left(y_{ij}|u_i^T v_j + r_{ij}, \sigma_y\right), \\ s_{ii'} &\sim Pr\left(s_{ii'}|u_i^T u'_{i'}\right). \end{aligned} \tag{28}$$

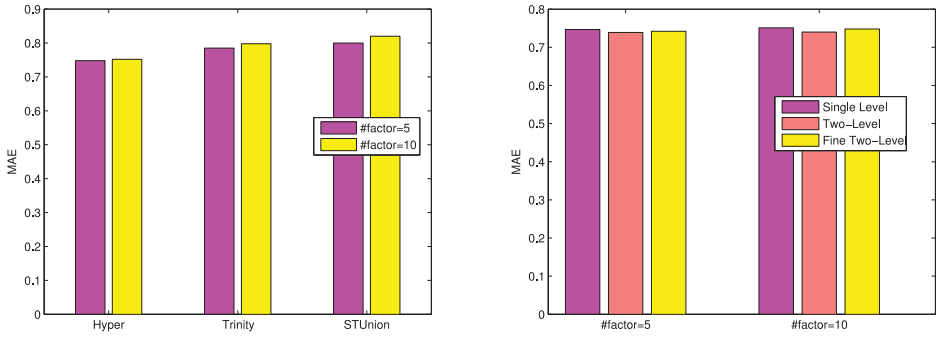
And FST/S is defined as

$$\begin{aligned} y_{ij} &\sim Pr\left(y_{ij}|u_i^T v_j + r_{ij}, \sigma_y\right), \\ t_{jj'} &\sim Pr\left(t_{jj'}|v_j^T v'_{j'}\right). \end{aligned} \tag{29}$$

FST/T is similar to SoRec in Ma et al. [2008], but it is imported with the explicit features of users and things. In addition, we excluded the social influence factors, namely, FST/S, which has the same structure as FST/T since our model FST holds a symmetrical structure. We implemented FST/T, FST/S, and FST, and conducted experiments to study their performance. We varied the training ratio as 10%, 20%, and 50%, and the number of factors is 5 and 10, respectively.

The results in Table VI show that the performance of FST/T drops when we take out the things correlations from FST. This validates our intuition that things' correlations do affect users' decision making process and behavior pattern when they use services offered by different things. Another finding is that FST/S consistently performs better than FST/T. In other words, things correlations has more influence than users' social relations. Comparing with traditional Internet resources like images or documents, physical things have a bigger impact on users' behavior because of their closeness with people and its nonduplicability. This is indeed a unique feature of IoT. For instance, one physical thing cannot be simultaneously used by multiple users, which will affect users' usage behavior. For example, if microwave in room1 is in use now, someone else has to look for another microwave in room2 even though he planned to use the microwave in room1 because it is closer to his office. Although this affects the things usage, we found that the improvement is not quite obvious. The reason might lie in that our dataset is not sparse due to its relatively small size. The impact of fusing things correlations on top of users' relations does not have significant impact on the recommendation accuracy.

4.2.3. Impact of Different Things Correlation Derivation. We compared our hypergraph-based approach with other conventional graph-based approaches, which are described as follows:



(a) (b)

Fig. 7. (a) Impact of hypergraph-based correlations derivation and (b) impact of single-level and two-level time decomposition.

—*Trinity*. This method constructs three separate graphs from things usage events, namely, *user-thing* graph, *time-thing* graph, and *location-thing* graph, respectively. We name this approach as *trinity* [Yao and Sheng 2012].
—*STUnion*. This method builds two graphs: one is using a *spatiotemporal* graph and a *social* graph. The two graphs model things usage contextual information and user-thing relationships, respectively. We name this approach as *STUnion* [Yao et al. 2013].

Figure 7(a) shows the significant improvement of the performance by using our proposed hypergraph-based approach. Our approach outperforms the other two conventional graph-based methods. The reason is that the hypergraph-based approach naturally integrates various relations into a unified framework without information loss, where high-order correlations across different entities are well captured, which encodes the users' preferences, contextual connections of things, and hence makes the results better. Compared with the hypergraph-based unified framework, the other two methods use conventional graphs, and squeezing the complex relations across heterogeneous entities causes some information loss, even though STUnion outperforms Trinity by integrating spatial information and temporal information together. Besides, mapping things usage information into separate graphs needs to tune parameters to find the optimal weight assignment of each graph. The optimal settings usually depend on different categories of things based on our previous study, which is another reason why the two conventional graph-based methods perform worse than the hypergraph-based method.

4.2.4. Impact of Time Granularity. To study the influence of different time granularities on system performance, we deployed three strategies of processing timestamps in our proposed framework. One strategy is *single-level decomposition*, in which the time was decomposed into 24 hours. Another strategy is more refined granularity, namely, *two-level time decomposition*, where we decomposed the times into discrete 24 hours and week/weekend days. Monday to 18:00 of Friday belong to the weekdays and 18:00 of Friday to Sunday belong to the weekend days. In this case, we have 24*2 time segments. The third strategy is, namely, *fine two-level decomposition*, where times were decomposed into 24 hours and 7 days, where we had 24*7 time nodes. Figure 7(b) shows the comparative results. The fine two-level decomposition slightly outperforms the other two strategies. The reason might be two-level strategy considers the human periodical behavior patterns. Intuitively, users behave differently when using different

things due to the nature of functions of things. As a result, different patterns, naturally formed in aggregated behaviors of visitors to various kinds of things, are embedded in the things usage logs. For instance, users generally tend to spend more time on things belonging to entertainment category or home category, and less time on office category on the weekend compared with weekdays. The second strategy nicely describes the features of periods.

5. RELATED WORK

In this section, we provide an overview about collaborative filtering approaches and thing search in the IoT, which are closely related to our approach.

5.1. Collaborative Filtering

Collaborative Filtering (CF) techniques exploit historical interactions of user behaviors for future prediction based on either neighborhood methods [Sarwar et al. 2001; Deshpande and Karypis 2004; Linden et al. 2003] or latent factor methods [Koren et al. 2009; Hofmann 2003; Salakhutdinov and Mnih 2007]. The idea behind neighborhood-based methods is that interactions between users and items can be inferred from the observation of users or items neighborhood. It predicts very well in learning the locality of dependencies, but does not explain the global relationships in the user-item interactions. The idea behind latent factor models is that preferences of a user are determined by a small number of latent factors, and it can learn the informative latent feature space but fails in capturing the local dependency of user or item neighborhood. Multiple unified models are proposed to combine neighborhood-based methods and latent factor based methods together. For example, Koren [2008] proposes an approach combining neighborhood-based CF and latent factor model together, and the performance is significantly improved. Salakhutdinov and Mnih [2007] proposes a probabilistic matrix factorization framework, which scales linearly with the number of observations and performs well on the large and sparse dataset. However, all of them cannot handle the cold-start problem very well. Agarwal and Chen [2009] addresses the cold-start problem by integrating explicit features of users and items into latent factors learning process. They develop a regression-based latent factor model for rating prediction, which uses features for factor estimation. In their methods, the user and item latent factors are estimated through independent regression on user and item features, and the recommendation is calculated from a multiplicative function on both user and item factors.

However, these research efforts assume users are independent or unrelated to each other. This assumption does not work well in the context of social networks where users social interactions have a big impact on recommending process. Most recent research efforts focus on exploiting the information of users' connections for recommendation. Many approaches have been developed to integrate users' social information in recommendation. Jamali and Ester [2009] proposes a social recommendation framework based on probabilistic matrix factorization via employing user social networks and user-item ratings. Yang et al. [2011a] designs a joint friendship and interest propagation model, where the user-item interest network and the user-user friendship network are jointly modeled through latent user and item factors. Zhou et al. [2012] proposes a kernel-based probabilistic matrix factorization, which incorporates external information into the matrix factorization process via assuming latent factors are in Gaussian distribution. Gu et al. [2010] develops a unified model for CF based on graph regularized weighted nonnegative matrix factorization. They adopt user demographic and item genre information to construct neighborhood graphs and incorporated user and item graphs in weighted nonnegative matrix factorization. Ma et al. [2008] proposes a matrix factorization method to exploit the social network information. Compared

with these research efforts, we construct the things correlation graph capturing global connections and similarities between things, and integrate the graph with users' social relation graph to learn the latent factors simultaneously.

5.2. Things Search

Searching related things (objects) is a key service in ubiquitous environments, such as the emerging IoT and smart environments. However, effectively searching for things is significantly more complicated than searching for documents because things are tightly bound to contextual information (e.g., location) and are often moving from one status to another. In addition, things do not have easily indexable properties, unlike human-readable text in the case of documents.

As in the traditional Web search, things search can be realized by exploiting keyword-based methods, like Microsearch [Tan et al. 2010], and Snoogle [Wang et al. 2008]. But the accuracy of these methods is not satisfying due to the insufficient content features of ubiquitous things. Another mainstream solution to search in a ubiquitous computing environment (e.g., IoT) is via semantic Web-related techniques. For example, Mietz et al. [2013] present a scalable semantic-based search model for the IoT. Perera et al. [2013] propose a middleware for sensor search based on users' priorities and other characteristics of sensors (e.g., reliability and accuracy). GSN [Aberer et al. 2006], Microsoft SensorMap [Nath et al. 2007], and linked sensor middleware [Le-Phuoc et al. 2011] support search for sensors based on textual metadata that describes the sensors (e.g., type and location of a sensor, measurement unit, object to which the sensor is attached). Such metadata is often manually entered by the person who deploys the sensors. Other users can then search for sensors with certain metadata by entering appropriate keywords. There are efforts to provide a standardized vocabulary to describe sensors and their properties such as SensorML[3] or the Semantic Sensor Network Ontology (SSN).[4] Unfortunately, these ontologies and their use are rather complex. It is problematic that end users are able to provide correct descriptions of sensors and their deployment context without the help from experts. In other words, this type of solutions require time-consuming prior and expertise knowledge, for example, define the descriptions of things and their corresponding characteristics under a uniform format such as Resource Description Framework (RDF). Furthermore, these solutions do not exploit the rich information about users historical interactions with things, containing implicit relations of different entities, for example, if some users have a similar usage pattern on some things, which may indicate close connection of these things.

Another alternative approach for searching things is based on search-by-example. The work in Truong et al. [2012] adopts this approach to sensors, that is, a user provides a sensor (respectively, a fraction of its past output as an example), and requests sensors that produced similar output in the past. Ostermaier et al. [2010] propose a real-time search engine for the Web of Things, which allows searching real-world entities having certain properties. They associate a Web page to a real-world entity (e.g., a meeting room) containing additional structured metadata about the sensors connected to it. This method takes care of the valuable information of historical data, but misses the relations among contextual information. Maekawa et al. [2012] propose a context-aware web search in ubiquitous sensor environments, and Yao and Sheng [2012] and Yao et al. [2013, 2014b] construct the models that capture the pairwise relations between things via mapping the contextual information into separate graphs. However, more complex relations between heterogeneous objects cannot be captured in these works.

[3]http://www.opengeospatial.org/standards/sensorml.
[4]http://www.w3.org/2005/Incubator/ssn/ssnx/ssn.

In addition, some useful information and structure might be lost when flatting the multidimensional information into graphs.

Compared with the related work mentioned previously, our proposed approach has two obvious advantages. First of all, the only data source needed in our approach is the interactions between users and things. These interaction events are handy to obtain with recent development of sensor networks and RFID technology. Secondly, we solve things searching problem from extracting the underlying relations among ubiquitous things and their interconnections via mining the rich information hidden in the human-thing interactions.

6. CONCLUSIONS AND FUTURE DIRECTIONS

Recommending the right thing to use at the specific time and location is a fundamental concern in the emergent IoT research. The work presented in this article is a continuing exploration of our previous work in Yao [2012] and Yao et al. [2013]. In particular, we have shown that both users' relations and things correlations have synergy effect on things recommendation. We propose a probabilistic matrix factorization based model integrating users' social relations (e.g., friendship) and things correlations together, where simultaneous latent factors are learned by revealing three dyadic relationships, including user-user, user-thing, and thing-thing relationships. Specifically, we develop a unified hypergraph to model complex relations of user-thing interactions for deriving latent things spatiotemporal correlations. We also integrate the content and other additional information of users and things to cope with the cold-start problem in recommendations. We conduct extensive experiments to validate and examine our proposed model, and the results demonstrate its feasibility and effectiveness. There are a number of interesting research directions that we are planning to work to further improve our approach:

—*Recommendation in Big Data*. The IoT is one of the biggest contributors to the emerging Big Data, in which vast amounts of data would need to be processed with more computational resources. There are two main directions to exploit our model in the Big Data environment. The first one is related to the mathematical properties of the algorithms (e.g., constructing more efficient data structures so that the data can fit in memory), and the second solution is related to the software engineering aspects of manipulating data on the scale of terabytes or even petabytes, such as constructing efficient, distributed, I/O infrastructure for accessing and manipulating the IoT data in a large-scale environment.

—*Online Update*. In the IoT, physical things are more dynamic compared with traditional Web resources. In another future work, we plan to improve our model so that it can adaptively propagate up-to-date information from things correlations network and make more accurate recommendations. Things states can be dynamically monitored and updated in the recommending process. It is important to achieve the capability of IoT recommendation in real time without sacrificing the accuracy.

REFERENCES

Karl Aberer, Manfred Hauswirth, and Ali Salehi. 2006. A middleware for fast and flexible sensor network deployment. In *Proceedings of the 32nd International Conference on Very Large Data Bases*. VLDB Endowment, 1199–1202.

Deepak Agarwal and Bee-Chung Chen. 2009. Regression-based latent factor models. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 19–28.

Sameer Agarwal, Kristin Branson, and Serge Belongie. 2006. Higher order learning with graphs. In *Proceedings of the 23rd International Conference on Machine Learning*. ACM, 17–24.

Jiajun Bu, Shulong Tan, Chun Chen, Can Wang, Hao Wu, Lijun Zhang, and Xiaofei He. 2010. Music recommendation by unified hypergraph: Combining social media information and music content. In *Proceedings of the International Conference on Multimedia*. ACM, 391–400.

Wei Chu and Seung-Taek Park. 2009. Personalized recommendation on dynamic content using predictive bilinear models. In *Proceedings of the 18th International World Wide Web Conference (WWW'09)*. 691–700.

Diane J. Cook and Maureen Schmitter-Edgecombe. 2009. Assessing the quality of activities in a smart environment. *Methods of Information in Medicine* 48, 5 (2009), 480.

Mukund Deshpande and George Karypis. 2004. Item-based top-n recommendation algorithms. *ACM Transactions on Information Systems* 22, 1 (2004), 143–177.

Quanquan Gu, Jie Zhou, and Chris H. Q. Ding. 2010. Collaborative filtering: Weighted nonnegative matrix factorization incorporating user and item graphs. In *Proceedings of the 2010 SIAM International Conference on Data Mining*. 199–210.

Thomas Hofmann. 2003. Collaborative filtering via Gaussian probabilistic latent semantic analysis. In *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*.

Mohsen Jamali and Martin Ester. 2009. TrustWalker: A random walk model for combining trust-based and item-based recommendation. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 397–406.

Tatsuya Kameda, Yohsuke Ohtsubo, and Masanori Takezawa. 1997. Centrality in sociocognitive networks and social influence: An illustration in a group decision-making context. *Journal of Personality and Social Psychology* 73, 2 (1997), 296.

Yehuda Koren. 2008. Factorization meets the neighborhood: A multifaceted collaborative filtering model. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 426–434.

Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix factorization techniques for recommender systems. *Computer* 42, 8 (2009), 30–37.

Danh Le-Phuoc, Hoan Nguyen Mau Quoc, Josiane Xavier Parreira, and Manfred Hauswirth. 2011. The linked sensor middleware: Connecting the real world and the semantic web. *Proceedings of the Semantic Web Challenge*.

Greg Linden, Brent Smith, and Jeremy York. 2003. Amazon.com Recommendations: Item-to-item collaborative filtering. *IEEE Internet Computing* 7, 1 (2003), 76–80.

Hao Ma, Haixuan Yang, Michael R. Lyu, and Irwin King. 2008. Sorec: Social recommendation using probabilistic matrix factorization. In *Proceedings of the 17th ACM Conference on Information and Knowledge Management*. ACM, 931–940.

Takuya Maekawa, Yutaka Yanagisawa, Yasushi Sakurai, Yasue Kishino, Koji Kamei, and Takeshi Okadome. 2012. Context-aware web search in ubiquitous sensor environments. *ACM Transactions on Internet Technology* 11, 3 (2012), 12.

Miller McPherson, Lynn Smith-Lovin, and James M. Cook. 2001. Birds of a feather: Homophily in social networks. *Annual Review of Sociology* (2001), 415–444.

Richard Mietz, Sven Groppe, Kay Römer, and Dennis Pfisterer. 2013. Semantic models for scalable search in the internet of things. *Journal of Sensor and Actuator Networks* 2, 2 (2013), 172–195.

Suman Nath, Jie Liu, and Feng Zhao. 2007. Sensormap for wide-area sensor webs. *Computer* 40, 7 (2007), 0090–0093.

Benedikt Ostermaier, K. Romer, Friedemann Mattern, Michael Fahrmair, and Wolfgang Kellerer. 2010. A real-time search engine for the web of things. In *Proceedings of the International Conference on Internet of Things*. IEEE, 1–8.

Siddika Parlak, Ivan Marsic, and Randall S. Burd. 2011. Activity recognition for emergency care using RFID. In *Proceedings of the 6th International Conference on Body Area Networks*. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 40–46.

Charith Perera, Arkady Zaslavsky, Peter Christen, Michael Compton, and Dimitrios Georgakopoulos. 2013. Context-aware sensor search, selection and ranking model for internet of things middleware. In *Proceedings of the 14th International Conference on Mobile Data Management (MDM'13)*, Vol. 1. IEEE, 314–322.

Ruslan Salakhutdinov and Andriy Mnih. 2007. Probabilistic matrix factorization. In *Proceedings of the 21st International Conference on Neural Information Processing Systems (NIPS'07)*.

Badrul Sarwar, George Karypis, Joseph Konstan, and John Reidl. 2001. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th International Conference on World Wide Web*. ACM, 285–295.

Quan Z. Sheng, Xue Li, and Sherali Zeadally. 2008. Enabling next-generation RFID applications: Solutions and challenges. *IEEE Computer* 41, 9 (2008), 21–28.

Geetika Singla, Diane J. Cook, and Maureen Schmitter-Edgecombe. 2010. Recognizing independent and joint activities among multiple residents in smart environments. *Journal of Ambient Intelligence and Humanized Computing* 1, 1 (2010), 57–63.

Rashmi R. Sinha and Kirsten Swearingen. 2001. Comparing recommendations made by online systems and friends. In *DELOS Workshop: Personalisation and Recommender Systems in Digital Libraries*, Vol. 106.

Liang Sun, Shuiwang Ji, and Jieping Ye. 2008. Hypergraph spectral learning for multi-label classification. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 668–676.

Chiu C. Tan, Bo Sheng, Haodong Wang, and Qun Li. 2010. Microsearch: A search engine for embedded devices used in pervasive computing. *ACM Transactions on Embedded Computing Systems* 9, 4 (2010), 43.

Cuong Truong, Kay Romer, and Kai Chen. 2012. Sensor similarity search in the web of things. In *Proceedings of the International Symposium on World of Wireless, Mobile and Multimedia Networks (WoWMoM'12)*. IEEE, 1–6.

Haodong Wang, Chiu Chiang Tan, and Qun Li. 2008. Snoogle: A search engine for the physical world. In *Proceedings of the 27th Conference on Computer Communications (INFOCOM'08)*. IEEE.

Yanbo Wu, Quan Z. Sheng, Damith Ranasinghe, and Lina Yao. 2012. PeerTrack: A platform for tracking and tracing objects in large-scale traceability networks. In *Proceedings of the 15th International Conference on Extending Database Technology (EDBT'12)*.

Shuang-Hong Yang, Bo Long, Alex Smola, Narayanan Sadagopan, Zhaohui Zheng, and Hongyuan Zha. 2011a. Like like alike: Joint friendship and interest propagation in social networks. In *Proceedings of the 20th International Conference on World Wide Web (WWW'11)*. ACM, 537–546.

Shuang-Hong Yang, Bo Long, Alexander J. Smola, Hongyuan Zha, and Zhaohui Zheng. 2011b. Collaborative competitive filtering: Learning recommender using context of user choice. In *Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'11)*. ACM, 295–304.

Lina Yao. 2012. A propagation model for integrating web of things and social networks. In *Service-Oriented Computing-ICSOC 2011 Workshops*. Springer, 233–238.

Lina Yao and Quan Z. Sheng. 2011. Particle filtering based availability prediction for web services. In *Service-Oriented Computing*. Springer, 566–573.

Lina Yao and Quan Z. Sheng. 2012. Exploiting latent relevance for relational learning of ubiquitous things. In *Proceedings of the 21st ACM International Conference on Information and Knowledge Management (CIKM'12)*. 1547–1551.

Lina Yao, Quan Z. Sheng, Byron Gao, Anne H. H. Ngu, and Xue Li. 2013. A model for discovering correlations of ubiquitous things. In *Proceedings of IEEE International Conference on Data Mining (ICDM'13)*.

Lina Yao, Quan Z. Sheng, Anne H. H. Ngu, Helen Ashman, and Xue Li. 2014b. Exploring recommendations in internet of things. In *Proceedings of the 37th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 855–858.

Lina Yao, Quan Z. Sheng, Anne H. H. Ngu, and Byron Gao. 2014a. Keeping you in the loop: Enabling web-based things management in the internet of things. In *Proceedings of the 23rd ACM International Conference on Information and Knowledge Management*. ACM.

Dengyong Zhou, Jiayuan Huang, and Bernhard Schölkopf. 2006. Learning with hypergraphs: Clustering, classification, and embedding. In *Advances in Neural Information Processing Systems*. 1601–1608.

Tinghui Zhou, Hanhuai Shan, Arindam Banerjee, and Guillermo Sapiro. 2012. Kernelized probabilistic matrix factorization: Exploiting graphs and side information. In *SDM*. 403–414.

Xiaojin Zhu. 2006. Semi-supervised learning literature survey. *Computer Science, University of Wisconsin-Madison* 2 (2006).

Received September 2014; revised October 2015; accepted October 2015