ThingsNavi: Finding Most-Related Things via Multi-Dimensional Modeling of Human-Thing Interactions

Lina Yao, Quan Z. Sheng, and Nickolas J.G. Falkner
School of Computer Science
The University of Adelaide
Adelaide, SA 5005, Australia
{lina, qsheng, jnick}@cs.adelaide.edu.au

Anne H.H. Ngu
Department of Computer Science
Texas State University
601 University Drive, San Marcos, USA
angu@txstate.edu

ABSTRACT

With the fast emerging Internet of Things (IoT), effectively and efficiently searching and selecting the most related things of a user's interest is becoming a crucial challenge. In the IoT era, human interactions with things are taking place at a new level in ubiquitous computing. These interactions initiated by humans are not completely random and carry rich contextual information. In this paper, we propose a things searching approach based on a hypergraph, called *ThingsNavi*, where given a target thing, other related things can be found by fully exploiting human-thing interactions in terms of multi-dimensional, contextual information (e.g., spatial information, temporal information, user identity). In particular, we construct a *unified hypergraph* to represent the rich structural and contextual information in human-thing interactions. We formulate the correlated things search as a ranking problem on top of this hypergraph, in which the information of target things can be propagated through the structure of the hypergraph. We evaluate our approach by using real-world datasets and the experimental results demonstrate its effectiveness.

Categories and Subject Descriptors

H.3.5 [**Information Storage and Retrieval**]: Online Information Services; H.4.0 [**Information Systems**]: General

Keywords

Internet of Things, things discovery, hypergraph, ranking

1. INTRODUCTION

The Internet of Things (IoT) will connect billions of things, which offers the capability of integrating the information from both the physical world and the virtual world. With IoT, it becomes possible to infer the status of real-world entities (i.e., things) with minimal delay [17, 22]. With many things connected and interacting over the Internet, there is an urgent need to effectively manage things, which is critical for a number of important applications such as things discovery (e.g., finding a quiet restaurant), recommendation (e.g., suggesting a device that can consume a video stream), and things mashup (e.g., composing device functionalities for a new service) [24, 25].

Given a target thing, searching its relevant counterparts still remains a challenge due to the specific natures of things in IoT. We summarize the main obstacles as follows:

- *Lack of uniform features*. Things are diverse and heterogeneous in terms of functionality, access methods or descriptions. Some things have meaningful descriptions while many others do not [4, 24]. As a result, it is quite challenging to discover the explicit relationships among heterogeneous things. Things cannot be easily represented in a meaningful feature space. They usually only have very short textual descriptions and lack a uniform way of describing the properties and the services they offer [7].
- *Lack of structural interconnections*. Correlations among things are not obvious and are difficult to discover. Unlike social networks of people, where users have observable links and connections, things often exist in isolated settings and the explicit interconnections between them are typically limited [23].

With the development of ubiquitous computing, and recently IoT, human-thing interactions can be easily recorded and obtained. These interactions are not completely random and carry rich information that can be mined and utilized [16]. 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 [10]. 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. For example, kitchenware is more frequently used during dining times and has higher likelihood to stay together in a kitchen or dining room.

In this paper, we propose a unified hypergraph to represent the various entities and heterogeneous relationships in collection of things usage events. A hypergraph is a generalization of the ordinary graphs, where its edges, also known as *hyperedges*, are arbitrary non-empty subsets of the vertex set [26, 2]. With our hypergraph modeling, the vertices represent various types of entities in human-thing interactions (e.g., people, things, temporal and spatial information etc.), and the hyperedges represent heterogeneous relations among entities via connecting arbitrary subsets of vertice. In this way, we not only incorporate heterogeneous relationships, e.g., the friendships between users or similarities between locations, but

MOBIQUITOUS 2014, December 02-05, London, Great Britain

DOI 10.4108/icst.mobiquitous.2014.258007

also utilize the users' historical interactions on things collections. Based on this unified hypergraph, we also develop an algorithm to rank related things according to a query by propagating the information of the target thing through the structure of this constructed hypergraph. The main contributions of this paper are three-fold:

- We integrate the multi-dimensional, contextual information of human-thing interactions into a unified hyper graph, including a thing and its identity information (users), temporal information (time stamps) and spatial information (location), without information loss.
- The only data sources used in our approach are things usage events, which are captured from users' interactions with things. They are comparatively easy to obtain with recent development of sensing technology and related ubiquitous techniques.
- We conduct comprehensive evaluations of our proposed approach using real-world datasets, including one dataset collected from our testbed and the CASAS datasets from Washington State University. The experimental results demonstrate the feasibility of our approach.

The remainder of the paper is organized as follows. We review some existing work related to this paper in Section 2. Section 3 defines related concepts and formulates the research problems. Section 4 describes the technical details of our *ThingsNavi* approach. We present our evaluation results in Section 5. Finally, some potential application scenarios of our proposed approach are highlighted in Section 6 and concluding remarks are given in Section 7.

2. RELATED WORK

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

Much like the traditional Web search, things search can be realized by exploiting keyword-based methods, like Microsearch [20], and Snoogle [22]. 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 is via semantic Web related techniques. For example, Mitetz et al. [11] present a scalable semantic-based search model for Internet of Things. Perera et al. [15] propose a middleware for sensor search based on users' priorities and other characteristics of sensors (e.g., reliability and accuracy). GSN [1], Microsoft SensorMap [12] and linked sensor middleware [8] 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 and then can be searched based on keywords. There are efforts to provide a standardized vocabulary to describe sensors and their properties such as SensorML[1] or the Semantic Sensor Network Ontology (SSN)[2]. Unfortunately, these ontologies and their use are rather complex. Moreover, it is problematic for end users to provide correct descriptions of sensors and their deployment context without the help from experts. In other words, these types of solutions require expertise knowledge, e.g., define the descriptions of things and their corresponding characteristics under a uniform format such as Resource Description Framework (RDF). Furthermore, none of these solutions considers the rich information derived from user's historical interactions with things.

Alternative approaches for searching things are based on search-by-example. The work in [21] adopts this approach to sensors, i.e., 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. [14] propose a real-time search engine for 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 exploits the information of historical data, but does not consider the relationships among contextual information of things. Maekawa et al. [9] propose a context-aware Web search in ubiquitous environments, and Yao et al. [23, 24] construct the models that capture the pairwise relations between things via mapping the contextual information into separate graphs. However, complex relationships between heterogeneous objects can not be captured in these approaches. In addition, some useful information may not be maintained during the graph construction process.

Compared with the related work mentioned above, our proposed approach, to be detailed next, 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 easy to obtain with recent development of sensor networks and radio frequency identification (RFID) technology. Secondly, we solve things searching problem by extracting the underlying relations among ubiquitous things and their interconnections. The underlying relations are captured via mining the rich information hidden in the human-thing interactions.

3. PROBLEM STATEMENT

The only data source used in our work is the interactions between humans and things, namely *things usage events*. Each event happens when a person interacts with a particular thing. Each usage event record is a quartuple of $<ThingID, UserID, Timestamp, Location>$ as described in the following.

DEFINITION 1 (THINGS USAGE EVENT). *Let $\mathcal{T} = \{t_1, \ldots, t_n\}$, $\mathcal{U} = \{u_1, ..., u_m\}$, $\mathcal{I} = \{i_1, ..., i_p\}$ and $\mathcal{L} = \{l_1, ..., l_q\}$ represent the set of things, users, timestamps and locations, respectively. A usage event of a thing t, denoted by $r \in \mathcal{R} = \{r_1, ..., r_i\} = \{< t, u, i, l > | t \in \mathcal{T} \wedge u \in \mathcal{U} \wedge i \in \mathcal{I} \wedge l \in \mathcal{L}\}$, indicates that user u used thing t located in a specific location l at timestamp i.*

Our goal is to model the various relationships in things usage events $\mathcal{R}$, and to make prediction of correlations amongst things, denoted as a query vector y_i. Our research goal can be formally formulated as Problem 1.

PROBLEM 1 (THINGS SEARCHING PROBLEM). *Given a hypergraph $HG(\mathcal{V}, \mathcal{E}, \mathcal{W})$, where $\mathcal{V}$ is the set of nodes and $\mathcal{E}$ is the set of hyperedges, $\mathcal{W}$ are the weights of hyperedges. Each node in $\mathcal{V}$ represents an entity in things usage events including users, time intervals, things and locations. Each hyperedge represents the relations between entities. Given a query node indicating thing $v \in \mathcal{V}$, our proposed method can provide a ranked list of nodes in $\mathcal{V}$ as the most relevant things measuring by relatedness scores.*

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

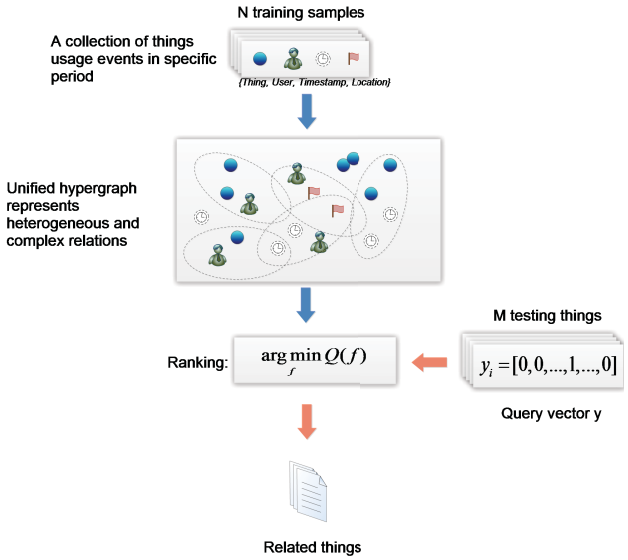


Figure 1: Overview of ThingsNavi

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

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

SUBPROBLEM 2 (RANKING). *Given the constructed hypergraph HG, learn a ranking function that can produce a list of related things for each target thing.*

Ideally, a solution should be *robust* in capturing and reflecting the hidden patterns in usage events, and also be *efficient* in ranking the candidates. In the rest of the paper, we will focus on introducing our proposed approach, *ThingsNavi*.

4. OUR APPROACH

Our proposed solution can be decomposed into two stages: i) the *descriptive stage* and ii) the *predictive stage*. The first stage aims at embracing the multiple relationships in things usage events and in particular, we 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 built at stage 1 (corresponding to Subproblem 2). In this section, we will describe our proposed approach in details and Figure 1 depicts the high level overview of ThingsNavi.

4.1 Subproblem 1: Modeling

We first introduce the motivations behind using a hypergraph to model things usage events, and then describe how to realize the modeling task defined in Subproblem 1.

4.1.1 The Intuitions

As mentioned before, a thing usage event is a four-element 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 such rich information from the usage events:

- *How to model heterogeneous relations with minimized information loss.* The various entities and relations make it difficult to develop a *unified* framework taking into account all things and relationships. For example, there are three types of contextual attributes in things usage events. To illustrate their relations with things, we generally need to construct three separate graphs, e.g., a user-thing graph, a location-thing graph, and a time-thing graph. However, dividing multiple relationships into separated graphs can cause non-trivial information loss.
- *How to model high-order relationships.* There exist some relations which are more complex, and it will cause information loss if simply being modeled using low-order modeling approach, e.g., using pairwise similarity based graphs or traditional multivariate regression methods. In addition, a pairwise relation between two nodes is only based on their own characteristics. If one of them is corrupted, their relation will be broken down as well. As a result, the relations between pairwise nodes are not stable.
- *How to reveal latent connections.* The complicate dependencies among a variety of entities make it hard to detect a latent relationship between entities. For example, although the usage frequency of things t_1 and t_3 being used together falls below some threshold, they might actually be meaningfully related. Let us say $<t_3, t_2>$ are used together frequently, and $<t_1, t_2>$ are also used frequently together. The presence of t_2 actually provides some possible latent connection for $<t_1, t_3>$, which needs to be uncovered.

Conventional graphs can only overcome these obstacles partially. 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 2 (a) shows a simple example to compare the scenarios of using a conventional graph (left) and of using a hypergraph (right) to model user interactions on things. In this example, both user u_1 and user u_2 access a thing t_1. The conventional graph obviously can not capture this connection. For example, to show user u_1 and u_2's relationships with t_1, we need two edges $<u_1, t_1>$ and $<u_2, t_1>$. We can not get the relationship between u_1 and u_2 on whether they access the same thing. However, the hypergraph can tackle this problem and fit this scenario naturally due to its edge can contain any number of arbitrary nodes. The hyperedge $< u1, u_2, t_1 >$ clearly demonstrates the relations among the entities. Figure 2 (b) shows one thing is highly possible to be accessed by multiple persons during a specific time frame. The conventional graph (left) describing binary correlations can not capture this high-order relations, on the other hand, the hypergraph (right) can accomplish this complex relations naturally.

4.1.2 Hypergraph Overview

As a generalization of the conventional graphs, a hypergraph can address challenges mentioned above (as demonstrated in Figure 2). 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. Hypergraphs have been extensively explored in recommendation [3] and multi-label classification [19].

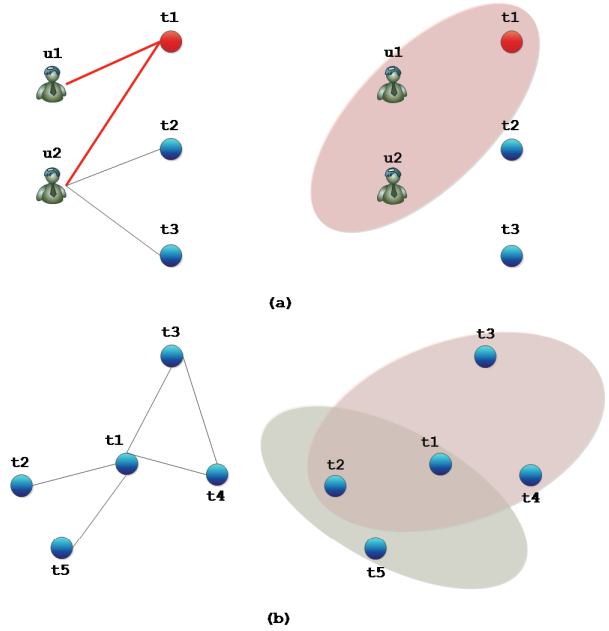


Figure 2: Illustrative example of hypergraph vs. conventional graph on modeling things usage events for unified relations and high-order relations: (a) the conventional graph (left) in which a pair of user and thing is connected if the user accessed this thing. This graph can not reflect whether the graph measures pairwise relations, e.g., $< u_1, t_1 >$ and $< u_2, t_1 >$, and the hypergraph (right), where each hyperedge can contain any number of arbitrary nodes to exhibit more complex relations, e.g., hyperedges can be written as $< u_1, u_2, t_1 >$ (ellipse shadow), from which we know both users $< u_1, u_2 >$ access t_1; (b) the conventional graph (left) in which two things can be connected if they are accessed by a same user, but it can not tell whether this thing is accessed by other users, and the hypergraph (right) can reasonably represents this complex relations.

Let $HG(\mathcal{V}, \mathcal{E}, \mathcal{W})$ be a weighted hypergraph with the vertex set $\mathcal{V} = \mathcal{U} \cup \mathcal{T} \cup \mathcal{L} \cup \mathcal{S}$ and the set of hyperedges $\mathcal{E}$. A hyperedge e is a non empty subset of $\mathcal{V}$ where $\cup_{e \in \mathcal{E}} = \mathcal{V}$, $\mathcal{W} : \mathcal{E} \rightarrow \mathbb{R}^+$ 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 $\{v_1, e_1, v_2, e_2, ..., e_{k-1}, v_k\}$ 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 [3, 26]:

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

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

$$\begin{aligned} d(v) &= \sum_{e \in \mathcal{E}} \mathcal{W}(e) h(v, e) \\ \delta(e) &= \sum_{v \in \mathcal{V}} h(v, e) = |e| \end{aligned} \tag{2}$$

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.

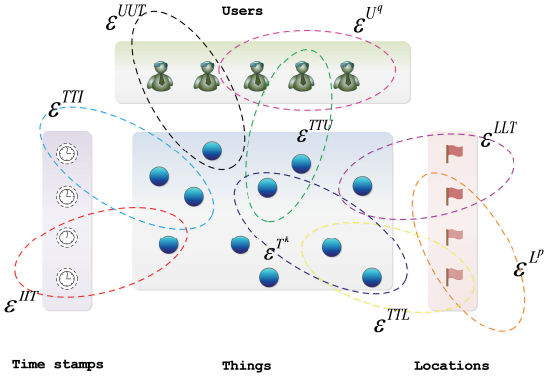


Figure 3: Illustration of relationships representation via constructing a hypergraph from things usage events. There are four types of entities in things usage events including things $\mathcal{T}$, users $\mathcal{U}$, locations $\mathcal{L}$ and timestamps $\mathcal{I}$. Nine types of relations include friendship relations $\mathcal{E}^{\mathcal{U}^q}$, location correlations $\mathcal{E}^{\mathcal{L}^p}$, things textual correlations $\mathcal{E}^{\mathcal{T}^k}$ relations of things usage with different contextual attributes, e.g., $\mathcal{E}^{\mathcal{TTU}}$, $\mathcal{E}^{\mathcal{TTI}}$, $\mathcal{E}^{\mathcal{TTL}}$, $\mathcal{E}^{\mathcal{UUT}}$, $\mathcal{E}^{\mathcal{IIT}}$, and $\mathcal{E}^{\mathcal{LLT}}$.

4.1.3 Unified Hypergraph Modeling

In this section, we describe the construction of a unified hypergraph to model the rich contextual information in things usage events. Figure 3 depicts how we map the entities and extracted relations as a unified hypergraph.

A hyperedge in our things usage hypergraph can be a set of vertices with either the same type or different types of relationships. The former kind of hyperedges capture the relationships among the same type of objects, while the latter capture the relationships across different types of objects. Things usage events consist of four types of entities. Let $\mathcal{U}$ denote the set of users, $\mathcal{T}$ 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* relationships across entities. Another two high-order relationships are constructed within two types of entities including users and locations, as well as one relationship being constructed based on the content similarity of two things, which we call *internal relationships*. Table 1 summarizes all the hyperedges modeled by our unified hypergraph. We briefly introduce them in the following.

External Relationships. There are six external relationships in our model:

- $\mathcal{E}^{\mathcal{TTU}}$: this relation represents the scenario that two things are used by a same person. The weight $w(e^{t_i t_j u_k})$ for this relation is set to the frequency that both things t_i, t_j are used by person u_k, denoted as:

$$w(e^{t_i t_j u_k}) = |\{(t_i, t_j, u_k)\}| t_i \in \mathcal{T}, t_j \in \mathcal{T}, u_k \in \mathcal{U}| \tag{3}$$

and the weight can be normalized as:

$$w(e^{t_i t_j u_k}) = \frac{w(e^{t_i t_j u_k})}{\sqrt{\sum_{l=1}^{|\mathcal{T}|} w(e^{t_l t_j u_k})} \sqrt{\sum_{m=1}^{|\mathcal{T}|} w(e^{t_i t_m u_k})}} \tag{4}$$

Table 1: Hyperedges Derived from Things Usage Events

External Relations across Entities		
Index	Notation	Hyperedges
1	$\mathcal{E}^{\mathcal{TTU}}$	Thing-Thing-User
2	$\mathcal{E}^{\mathcal{TTI}}$	Thing-Thing-Time
3	$\mathcal{E}^{\mathcal{TTL}}$	Thing-Thing-Location
4	$\mathcal{E}^{\mathcal{UUT}}$	User-User-Thing
5	$\mathcal{E}^{\mathcal{LLT}}$	Location-Location-Thing
6	$\mathcal{E}^{\mathcal{ITT}}$	Time-Time-Thing
Internal Relations within Entities		
Index	Notation	Hyperedges
7	$\mathcal{E}^{\mathcal{L}^p}$	p-nearest locations
8	$\mathcal{E}^{\mathcal{U}^q}$	q-nearest users
9	$\mathcal{E}^{\mathcal{T}^k}$	k-nearest things

- $\mathcal{E}^{\mathcal{TTI}}$. This represents the scenario that two things are used in 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 a same time period i_k. The calculation is same as Equation 4.

- $\mathcal{E}^{\mathcal{TTL}}$. This represents that two things are used in 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 same as Equation 4.

- $\mathcal{E}^{\mathcal{UUT}}$. In this relation, if one thing is used by two users, we assign the weight to 1.

- $\mathcal{E}^{\mathcal{ITT}}$. In this relation, if one thing is used at a same timestamp, we assign the weight to 1.

- $\mathcal{E}^{\mathcal{LLT}}$. In this relation, if one thing is used in a same location, we assign the weight to 1.

Internal Relationships. There are also three internal relationships in our model:

- $\mathcal{E}^{\mathcal{L}^p}$. 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:

$$w(e^{l_i^p}) = \frac{1}{p}\sum_{j=1}^{p} sim(l_i, l_j) \tag{5}$$

  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:

$$sim(l_i, l_j) = \frac{|\Gamma_i^o \cap \Gamma_j^o|}{|\Gamma_i^o \cup \Gamma_j^o|} \tag{6}$$

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

- $\mathcal{E}^{\mathcal{U}^q}$. In this relation, we consider the similarity between users. In the hypergraph, a hyperedge of this type represents each user and her top q similar users. For each user u_i, the weight of hyperedge is calculated using:

$$w(e^{u_i^q}) = \frac{1}{k}\sum_{j=1}^{q} sim(u_i, u_j) \tag{7}$$

  where $sim(u_i, u_j)$ is the similarity between two users, which is influenced by the social links between users, reflecting the homophily theory meaning that similar users may have similar interests. We use the cosine similarity to calculate the similarity between two users as follows:

$$sim(u_i, u_j) = \frac{e^{\alpha cos(b(u_i), b(u_j))}}{\sum_{c \in \Omega(u_i)} e^{\alpha cos(b(u_i), (b(u_c))}} \tag{8}$$

  where $cos(b(u_i), b(u_j)) = \frac{b(u_i) \cdot b(u_j)}{||b(u_i)||||b(u_j)||}$, $\Omega(u_i)$ is the set of the user u_i's friends (i.e., $u_j \in \Omega(u_i)$), $b(u_i)$ is the binary vector of things used by user u_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 interacted with the same things.

- $\mathcal{E}^{\mathcal{T}^k}$. Although things descriptions are limited[3], to fully utilize the available information, we extract bag-of-keywords from the descriptions. By analyzing the textual descriptions, it is possible to extract the most common terms that represent the corresponding thing (i.e., content-based features). It should be noted that the common implementation of TF/IDF gives equal weights to the term frequency and inverse document frequency (i.e., $w = tf \times idf$). We choose to give higher weight to the idf value (i.e., $w = tf \times idf^2$). The reason behind this modification is to normalize the inherent bias of the tf measure in short documents. Traditional TF/IDF applications are concerned with verbose documents (e.g., books, articles, and human-readable Web pages). However, documents describing things are relatively short. Therefore, the frequency of a word within a document tends to be incidental, and the document length component of the TF generally has little or no influence.

  The weight of each thing in the content-level is the averaged similarity between the querying thing and the other things, which can be computed using:

$$w(e_r^{t_i}) = \frac{1}{k}\sum_{j=1}^{k} sim(t_i, t_j) \tag{9}$$

  where $sim(t_i, t_j)$ is calculated using cosine similarity as follows:

$$sim(t_i, t_j) = \frac{(t_i) \cdot (t_j)}{||t_i||||t_j||} \tag{10}$$

Based on hypergraph modeling introduced above, we can derive the vertex-hyperedge incidence matrix $\mathcal{H}$ (as shown in Table 2) 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.

4.2 Subproblem 2: Ranking

In this section, we describe our approach for ranking the related things, which is designed to solve Subproblem 2.

[3]In our implementation, ubiquitous things are exposed as RESTful Web services, and each of them has a short web description.

Table 2: The incident matrix $\mathcal{H}$ of the proposed hypergrah

	$\mathcal{E}^{\mathcal{TTU}}$	$\mathcal{E}^{\mathcal{TTI}}$	$\mathcal{E}^{\mathcal{TTL}}$	$\mathcal{E}^{\mathcal{UUT}}$	$\mathcal{E}^{\mathcal{LLT}}$	$\mathcal{E}^{\mathcal{ITT}}$	$\mathcal{E}^{\mathcal{L}^p}$	$\mathcal{E}^{\mathcal{U}^q}$	$\mathcal{E}^{\mathcal{T}^k}$
$\mathcal{T}$	$\mathcal{TE}^{\mathcal{TTU}}$	$\mathcal{TE}^{\mathcal{TTI}}$	$\mathcal{TE}^{\mathcal{TTL}}$	$\mathcal{TE}^{\mathcal{UUT}}$	$\mathcal{TE}^{\mathcal{LLT}}$	$\mathcal{TE}^{\mathcal{ITT}}$	0	0	$\mathcal{TE}^{\mathcal{T}^k}$
$\mathcal{U}$	$\mathcal{UE}^{\mathcal{TTU}}$	0	0	$\mathcal{UE}^{\mathcal{UUT}}$	0	0	0	$\mathcal{UE}^{\mathcal{U}^q}$	0
$\mathcal{I}$	0	$\mathcal{IE}^{\mathcal{TTI}}$	0	0	0	$\mathcal{IE}^{\mathcal{ITT}}$	0	0	0
$\mathcal{L}$	0	0	$\mathcal{LE}^{\mathcal{TTL}}$	0	$\mathcal{LE}^{\mathcal{LLT}}$	0	$\mathcal{LE}^{\mathcal{L}^p}$	0	0

To infer the related things for a given querying thing, we adopt a spectral clustering based semi-supervised learning framework [27]. The goal is that given the query thing t_i, to estimate a ranking function $\mathbf{f}$ which allocates other things different relatedness scores $f(i)$ over the hypergraph HG, indicating their relevance to the querying thing $t_i \in \mathcal{V}$. Let $y_i \in \mathbf{y}$ be the query vector for thing t_i. The learning process can be formulated under a regularization framework as follows:

$$Q(\mathbf{f}) = E(\mathbf{f}) + \mu\ell(\mathbf{f}, \mathbf{y}) \tag{11}$$

where $E(\mathbf{f})$ is the quadratic energy function, which smooths the vertices that are nearby according to Markov assumption, in other word, the nearby vertices on HG should have similar relatedness scores, $\ell(\mathbf{f}, \mathbf{y})$ is the loss function measuring the difference between predicted relatedness score and true scores. μ is the regularizer. The equation can be expanded as:

$$\begin{aligned} Q(\mathbf{f}) =& \frac{1}{2}\sum_{i,j=1}^{|\mathcal{V}|}\sum_{e\in\mathcal{E}}\frac{1}{\delta(e)}\sum_{(v_i,v_j)\subset e}\mathcal{W}(e)\left|\frac{\mathbf{f}_i}{\sqrt{d(v_i)}} - \frac{\mathbf{f}_j}{d(v_j)}\right|^2 \\ &+ \mu\sum_{i=1}^{|V|}||\mathbf{f}_i - y_i||^2 \end{aligned} \tag{12}$$

The optimal relatedness scores can be learned recursively by solving the optimization problem in a closed form [26]:

$$\begin{aligned} &\mathbf{f}^* = \text{argmin}_{\mathbf{f}} Q(\mathbf{f}) = (1-\alpha)(1-\alpha\Theta)^{-1}\mathbf{y} \\ &\text{where} \\ &\alpha = 1/1+\mu \text{ and } \Theta = \mathcal{D}_v^{-1/2}\mathcal{HW}\mathcal{D}_e^{-1}\mathcal{H}^T\mathcal{D}_v^{-1/2} \end{aligned} \tag{13}$$

Equation 13 can be reformulated as:

$$\mathbf{f}^* = \alpha\Theta\mathbf{f}^* + (1-\alpha)\mathbf{y} \tag{14}$$

The detailed algorithm of our *ThingsNavi* approach for things searching is summarized in Algorithm 1.

Algorithm 1: *ThingsNavi*

Input: A sequence of things usage events $\mathcal{H}$ and query y_i
Output: Related things ranking list $\{y_1, y_2, ..., y_n\}$

Constructing hypergraph $HG(\mathcal{V}, \mathcal{E}, \mathcal{W})$ as described in Section 4.1;
Computing incident matrix $\mathcal{H}$;
Computing weight matrix $\mathcal{H}$;
Computing vertice degree matrix $\mathcal{D}_v$;
Computing hyperedge degree matrix $\mathcal{D}_e$;
Computing $\Theta = \mathcal{D}_v^{-1/2}\mathcal{HW}\mathcal{D}_e^{-1}\mathcal{H}^T\mathcal{D}_v^{-1/2}$;
Initializing relatedness score vector y_i corresponding to t_i as: the i-th element is $y_{i_i} = 1$, other elements are equal to 0;
Let $\mathbf{f}^0 = y_i$;
while $\delta < \xi$ **do**
    $\mathbf{f}^{j+1} = \alpha\Theta\mathbf{f}^j + (1-\alpha)y_i$;
    for *k = 1:n* **do**
        $\delta \leftarrow \max|\mathbf{f}_k^{j+1} - \mathbf{f}_k^j|$;
    end
end
Output $\{y_1, ..y_m\}$ sorted by relatedness score with y_i.

4.3 Complexity Analysis

Clearly, in our ThingsNavi approach, there are two main processes that are responsible for computational complexity. One is the hypergraph construction process during the training phase, and the other is the searching and ranking process during the application phase. We recall that the external hyperedges reflect the pairwise relationships and similarities between two nodes while the internal hyperedges characterize the neighboring information on nodes. For the construction of internal relationships of the hypergraph, e.g., given n nodes consisting of users and locations, we need n^2 similarity calculations. So the time complexity is $\mathcal{O}(n^2)$ in order to construct the pairwise hypergraph. Similarly, for the construction of external relationships of the hypergraph, given m nodes, it needs m^2 similarity calculations. Thus its construction complexity is $\mathcal{O}(m^2)$. Therefore, considering two kinds of hyperedges in this work, the final similarity incidence matrix $\mathcal{H}$ with overall number of N entities has $\mathcal{O}(N^2)$ complexity.

For the time complexity of our semi-supervised search and rank algorithm, since our framework is inherited from graph-based SSL (semi-supervised learning), in which the weighted hypergraph is formed over labeled and unlabeled data (unlabeled data is the query thing in this work, labeled data is the training data), the unlabeled points will be assigned similarity scores based on the labels of their neighbors. This learning process is typically slow and its time complexity is $\mathcal{O}(N^3)$. Overall, the complexity of our proposed approach is $\mathcal{O}(N^3)$.

Currently, the validation of ThingsNavi is mainly conducted in a small-scale testbed (see next section for details). The dataset collected is not a large one and the performance of the approach is reasonable. For example, the total number of nodes including things, timestamps, locations and users of our dataset is 261, and the time used for completing a query is about 11.07 seconds in a typical desktop computer. It should be noted that our main focus of this paper is to demonstrate the potential of using a hypergraph to model the contextual relationships in human-thing interactions, which is impossible to be captured by using ordinary graphs. Scalability and performance improvement of ThingsNavi, which are important issues for the approach to be applied in large scale IoT environment, are the main concern of our future work. So far, we have already explored some possible techniques (e.g., using the hashing function, batch tuning process) to accelerate the searching process and to reduce the computational cost.

(a)

(b)

Figure 4: (a) some sensors and RFID devices; (b) sensor enabled microwave oven

5. THE EVALUATION

In this section, we report the experimental studies on evaluating the proposed *ThingsNavi* approach. We will first describe the data collection and the experimental settings, and then report various experimental results.

5.1 Data Collection

To validate our approach, we set up a testbed in the first author's home where approximate 60 physical things (e.g., cups, laptop, microwave oven, fridge, kettle, toaster) are monitored by attaching RFID tags and/or sensors. Figure 4(a) shows some devices used in our experiment including RFID readers, sensors and tags, and Figure 4(b) shows an example of a sensor-enhancement microwave oven in our testbed.

Figure 5 depicts the overall architecture of our testbed. When users interact with things, the *Event Detector* module records the events which are captured, e.g., by RFID readers or inferred based on sensor values. The related contextual information can be attached to each event (e.g., location information calculated using the localization algorithm, to be discussed later). Ten volunteers participated in the data collection phase by interacting with RFID tagged things for a period of four months, generating 20,179 records on the interactions of the things tagged in the experiments. More details of data acquisition can be found in [24].

Our dataset was augmented by using Washington State University's CASAS datasets[4], which was collected from a smart home environment. In particular, we used dataset1 [5] and dataset2 [18]. From the datasets, we identified 52 more things (e.g., bowl, door, coffee table and watertap) 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, we therefore mapped the location of things usage events related to L-11 to L-15

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

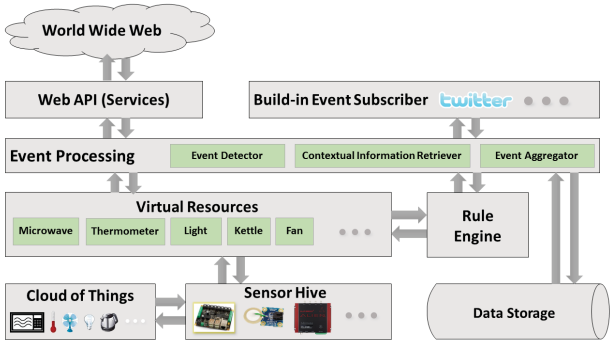


Figure 5: The architecture of our testbed

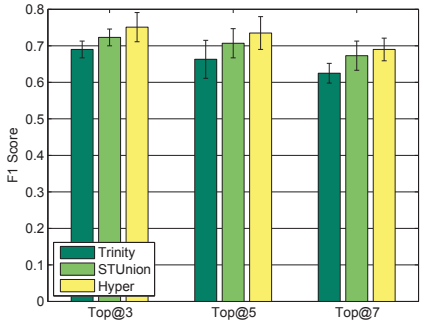
to the bathroom. Each activity can then be transformed as things usage events, to be used in our experiments. For example, consider the activity of "reading a magazine in couch". For this record, we converted it into two events, $<magazine, person1, timestamp, livingroom>$ and $<couch, person1, timestamp, livingroom>$. In summary, there were 261 entities used in the experiments including 179 things from six categories (e.g., entertainment, cooking etc), 24 time intervals, 37 locations, and 21 users.

Each usage event is associated with three main pieces of information: identity (user), temporal (timestamp) and spatial (location) information. To obtain user information, in our experiments, we used a manual labeling method where each participant needs to mark and record their activities. For the temporal information, we chose to split one day into 24 equal intervals (clusters). Each interval is one hour. For example, if the timestamp of a thing usage event is 9:07am, it will be assigned into the temporal cluster between 9:00am to 10:00am.

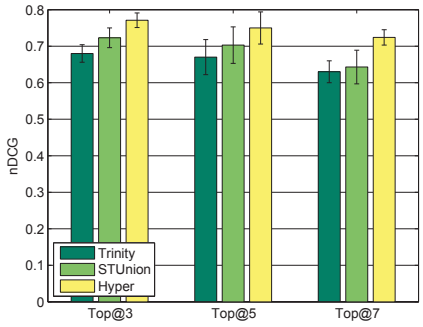
To get the localization information, we consider two situations. The first one is *static* things (e.g., refrigerator, microwave oven). The location information of such things are prior knowledge. The other situation is *mobile* things (e.g., RFID-tagged coffee mug). For such things, we provide a coarse-grain or fine-grain location information. For the coarse-grain method, since the Received Signal Strength Indication (RSSI) signal received from a tagged thing reveals its proximity to an RFID reader antenna. We divided an area in multiple zones and each zone is covered with a mutually exclusive set of RFID antennas. The zone scanned by the antenna with the maximum RSSI is taken to be the thing's location. For the fine-grain method, it is determined by comparing the signal descriptors from a thing at unknown location to a previously constructed radio map or fingerprints. We used the Weighted k Nearest Neighbors algorithm (w-kNN), where we find the most similar fingerprints and compute a weighted average of their 2D positions to estimate the unknown tag location [13]. For details, interested readers can refer to our previous work [24, 23].

5.2 Evaluation Settings and Metrics

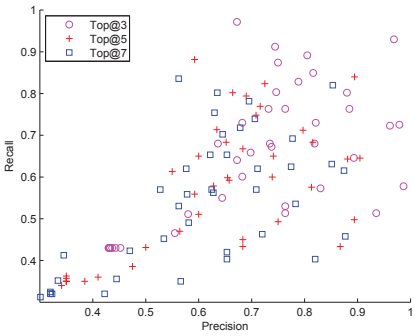
We selected 70% things usage events as the training set, and the rest as the testing set. The training set is used to build the hypergraph and learn the ranking function, and the testing dataset is used as the ground-truth dataset to evaluate the performance of our ThingsNavi approach. For each thing in the testing set, our approach produces top@x (x = 3,5,7) related things according to relatedness scores with the query thing. We measured how many things in the top@x results are related to the query thing using the metrics of F1-score and nDCG (the normalized Discounted Cumulative Gain).



(a)



(b)



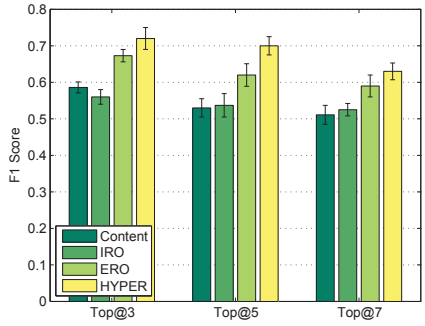
(c)

Figure 6: Performance comparison with hypergraph and conventional graph construction on top@3, top@5 and top@7: (a) F1 score, (b) nDCG and (c) precision-recall

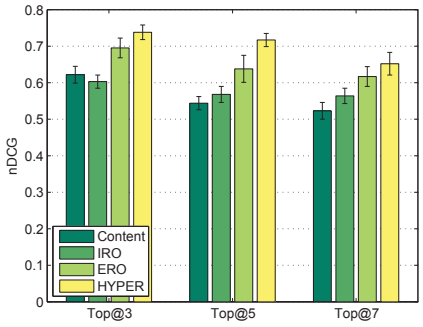
F1 is the harmonic mean of $Precision_x$ and $Recall_x$, which can be calculated as:

$$F1_x = \frac{2 \cdot Precision_x \cdot Recall_x}{Precision_x + Recall_x} \tag{15}$$

where $Precision_x = \frac{|Rel \cap Rec_x|}{|Rec_x|}$ and $Recall_x = \frac{|Rel \cap Rec_x|}{|Rec|}$, Rel is the related things (we assume the things with same labels are in a relevant set), and Rec_x is the set of top@x things output by ThingsNavi. Considering that people tend to select only the top few things, an approach with high top@x precision values is desirable in practice. $Precision_x$ measures how often is the related things actually present when the approach predicts it is. $Recall_x$ measures how many actually related things are recognized by the approach. For the ranking position in the results, we also adopted



(a)



(b)

Figure 7: Performance comparison with different combinations of hyperedges on top@3, top@5 and top@7: (a) F1 score and (b) nDCG

the normalized Discounted Cumulative Gain (nDCG) as the metric, which is the normalized position-discounted precision score. It gives more credit to the top-ranked things and is defined as the following:

$$DCG_x = \sum_{i=1}^{x} \frac{2^{rel_i} - 1}{\log_2(1 + p_i)} \tag{16}$$

where p_i is the ranking position of the i^{th} thing in the top@x things list, and rel_i is the relatedness score of the i^{th} thing at position p_i.The normalized DCG can be calculated using:

$$nDCG_x = DCG_x / IDCG \tag{17}$$

where $IDCG$, also called Ideal DCG, is calculated by sorting things of a result list by the relevance, producing the maximum possible DCG till position p_i. The range of IDCG is between $[0, 1]$.

5.3 Performance Comparison

Things searching is a new research area and there is very little work in the literature. We compared our hypergraph-based approach (i.e., ThingsNavi) with other conventional graph based approaches, which are described as follows:

- *Trinity*. This method constructs three separate graphs from things usage events, namely a *user-thing* graph, a *time-thing* graph, and a *location-thing* graph respectively. We name this approach as *Trinity* [23].
- *STUnion.* This method builds two graphs: a *spatio-temporal* graph and a *social* graph. The two graphs model things usage

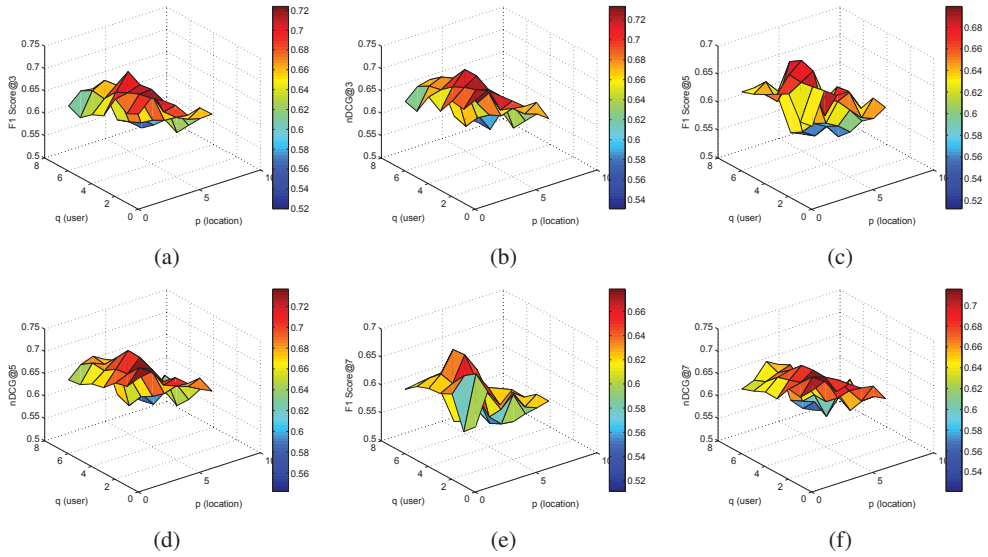


Figure 8: Performance with different combination of p with q: (a) and (b) are F1 and nDCG on top@3, (c) and (d) are F1 and nDCG on top@5, (e) and (f) are F1 and nDCG on top@7

contextual information and user-thing relationships respectively. We name this approach as *STUnion* [24].

Figure 6 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 a hypergraph-based approach naturally integrates various relationships into a unified framework without information loss, and the high-order correlations across different entities are well-captured, including users' preference, as well as the contextual connections of things. Compared to the hypergraph based unified framework, the other two methods use conventional graphs, which squeeze the complex relations across heterogeneous entities and cause some information loss. We also notice that STUnion outperforms Trinity because it integrates spatial information and temporal information together.

5.4 Impact on Hyperedges of Relations

In our proposed approach, we incorporate nine different hyperedges, indicating different relationships across or within heterogeneous entities to construct the hypergraph. In this way, the hypergraph can represent the complex relationships encoded in things usage events. To evaluate the impact of different relations over the hypergraph, we conducted experiments using several different combinations of hyperedges for hypergraph construction.

The first category of relationships including hyperedges index 1, index 4 and index 8 in Table 1, which encodes the users access patterns on things along with internal user relations and things' characteristics. We name these combined relations as IRO. Another category of relationships includes hyperedges index 2, index 3, index 5, index 6, and index 7, which are considered to decode the things similarity in terms of contextual dependencies. We name it as ERO. The third category considers hyperedges index 9, which represents textual-level similarity between things. We name it as the pure content-based approach (*Content* for short). The fourth category, HYPER, includes all the relationships.

The results are shown in Figure 7. It is obvious that the hypergraph (HYPER) outperforms the other three combinations. The reason is that HYPER considers multiple relationships between different entities, which fully makes use of the rich information hidden in these relations. It is noted that ERO outperforms IRO, which reflects that the contextual factors play a more dominant role in finding the desirable things.

5.5 Parameters Tuning

In our proposed method, we need to determine the values of p and q (see Equation 5 and 7 in Section 4.1) for locations and users, which indicate the number of nearest neighbors for locations and users respectively. To examine the effects of these parameters, we evaluated different combinations of $p \in [1, 7]$ and $q \in [1, 7]$ with F_1 and $nDCG$ on top@3, top@5 and top@7. Figure 8 shows that the performance reaches the best with p=3 and q=4. We also observe that p has bigger impact on the performance compared to q. The possible reason is due to the close dependence between things and their locations, e.g., kitchenware is mostly in the kitchen area.

6. APPLICATION SCENARIOS

Our model can be generalized and applied to many applications in mobile and ubiquitous environments. Here, we briefly describe some application domains to show the usefulness of our approach.

Human Activity Recognition. Our proposed approach provides a new angle to learn human activities in the ubiquitous environments. For example, we can set the sensors, RFID tags, their corresponding values and timestamps as nodes, to build a hypergraph, where some clustering algorithms (e.g., spectral clustering, modularity-based clustering) or proximity learning algorithm (e.g., random walk) [6] can be used to build the connections between users' activities and sensing values.

Location-based Services. Things usage events have similar structure with check-in records in location-based services, which usually comprise of <user, place, timestamp>. Our approach can be easily adapted to this area by mapping different data sources in the check-in records (e.g., user, place and timestamps) via hypergraph and then inferring human mobility patterns or landmark locations via selecting different nodes on the hypergraph for querying.

Healthcare. Our approach can contribute to the healthcare domain because the medical equipment/resource usage records are an important data source in healthcare. For example, it is possible to detect any abnormal medical equipment usage behaviors by e.g., detecting whether "left-ins" happen (e.g., glove left inside patient) after a surgery.

7. CONCLUSION

Recommending the right things to use at a specific time and location is a fundamental concern in the emerging Internet of Things (IoT) research and development. In this paper, we propose to solve this problem by mining things usage events. Our approach, named *ThingsNavi*, exploits a hypergraph to encode various relationships, forming a robust representation of things usage events without information loss. A ranking algorithm is also developed for recommending the most related things to a given specific thing (i.e., querying thing). The experimental results using real-world datasets demonstrate the effectiveness of the proposed approach.

There are several main directions for our future research. The first one centers on dealing with the dynamism of things. In real situations, physical things are more dynamic compared with traditional Web resources (e.g., movement of things). We plan to improve our model that can adaptively propagate up-to-date information from the hypergraph and make more accurate recommendations. The second one focuses on refining our model for pattern identification from human-thing interactions by considering specific contextual conditions. Finally, we also plan to evaluate our approach using larger-scale datasets.

8. REFERENCES

[1] K. Aberer, M. Hauswirth, and A. Salehi. A middleware for fast and flexible sensor network deployment. In *Proc. of the 32nd Intl. Conference on Very Large Data Bases*, pages 1199–1202. VLDB Endowment, 2006.

[2] S. Agarwal, K. Branson, and S. Belongie. Higher order learning with graphs. In *Proc. of the 23rd Intl. Conference on Machine Learning*, pages 17–24. ACM, 2006.

[3] J. Bu et al. Music recommendation by unified hypergraph: combining social media information and music content. In *Proc. of the Intl. Conference on Multimedia*, pages 391–400. ACM, 2010.

[4] B. Christophe, V. Verdot, and V. Toubiana. Searching the Web of Things. In *Proc. of the 5th IEEE Intl. Conf. on Semantic Computing*, pages 308–315, 2011.

[5] D. Cook, M. Schmitter-Edgecombe, et al. Assessing the quality of activities in a smart environment. *Methods of information in medicine*, 48(5):480, 2009.

[6] Y. Fujiwara, M. Nakatsuji, M. Onizuka, and M. Kitsuregawa. Fast and exact top-k search for random walk with restart. *Proceedings of the VLDB Endowment*, 5(5):442–453, 2012.

[7] T. Kindberg et al. People, places, things: Web presence for the real world. *Mobile Networks and Applications*, 7(5):365–376, 2002.

[8] D. Le-Phuoc, H. N. M. Quoc, J. X. Parreira, and M. Hauswirth. The linked sensor middleware–connecting the real world and the semantic web. *Proceedings of the Semantic Web Challenge*, 2011.

[9] T. Maekawa, Y. Yanagisawa, Y. Sakurai, Y. Kishino, K. Kamei, and T. Okadome. Context-aware web search in ubiquitous sensor environments. *ACM Transactions on Internet Technology (TOIT)*, 11(3):12, 2012.

[10] M. McPherson, L. Smith-Lovin, and J. Cook. Birds of a Feather: Homophily in Social Networks. *Annual Review of Sociology*, 27(1):415–444, 2001.

[11] R. Mietz, S. Groppe, K. Römer, and D. Pfisterer. Semantic models for scalable search in the internet of things. *Journal of Sensor and Actuator Networks*, 2(2):172–195, 2013.

[12] S. Nath, J. Liu, and F. Zhao. Sensormap for wide-area sensor webs. *Computer*, 40(7):0090–93, 2007.

[13] L. Ni, Y. Liu, Y. Lau, and A. Patil. LANDMARC: Indoor Location Sensing Using Active RFID. *Wireless Networks*, 10(6):701–710, 2004.

[14] B. Ostermaier, K. Romer, F. Mattern, M. Fahrmair, and W. Kellerer. A real-time search engine for the web of things. In *Internet of Things (IOT), 2010*, pages 1–8. IEEE, 2010.

[15] C. Perera, A. Zaslavsky, P. Christen, M. Compton, and D. Georgakopoulos. Context-aware sensor search, selection and ranking model for internet of things middleware. In *Proc. of IEEE 14th Intl. Conference on Mobile Data Management (MDM 2013)*, 2013.

[16] M. Philipose, K. Fishkin, M. Perkowitz, D. Patterson, D. Fox, H. Kautz, and D. Hahnel. Inferring activities from interactions with objects. *Pervasive Computing, IEEE*, 3(4):50–57, 2004.

[17] Q. Sheng et al. Ubiquitous RFID: Where are we? *Information Systems Frontiers*, 12(5):485–490, 2010.

[18] G. Singla, D. J. Cook, and M. Schmitter-Edgecombe. Recognizing independent and joint activities among multiple residents in smart environments. *Journal of Ambient Intelligence and Humanized Computing*, 1(1):57–63, 2010.

[19] L. Sun, S. Ji, and J. Ye. Hypergraph spectral learning for multi-label classification. In *Proc. of the 14th ACM SIGKDD Intl. Conference on Knowledge Discovery and Data Mining*, 2008.

[20] C. C. Tan, B. Sheng, H. Wang, and Q. Li. Microsearch: A search engine for embedded devices used in pervasive computing. *ACM Transactions on Embedded Computing Systems (TECS)*, 9(4):43, 2010.

[21] C. Truong, K. Romer, and K. Chen. Sensor similarity search in the web of things. In *World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2012 IEEE International Symposium on a*, pages 1–6. IEEE, 2012.

[22] H. Wang, C. C. Tan, and Q. Li. Snoogle: A search engine for the physical world. In *Proc. of the 27th Conference on Computer Communications (INFOCOM)*. IEEE, 2008.

[23] L. Yao and Q. Z. Sheng. Exploiting latent relevance for relational learning of ubiquitous things. In *Proc. of the 21st ACM Intl. Conference on Information and Knowledge Management (CIKM 2012)*, 2012.

[24] L. Yao, Q. Z. Sheng, B. Gao, A. H. H. Ngu, and X. Li. A Model for Discovering Correlations of Ubiquitous Things. In *Proc. of IEEE Intl. Conference on Data Mining (ICDM 2013)*, 2013.

[25] L. Yao et al. Exploring recommendations in internet of things. In *Proc. of the 37th Intl. ACM SIGIR Conference on Research & Development in Information Retrieval*, 2014.

[26] D. Zhou, J. Huang, and B. Schölkopf. Learning with hypergraphs: Clustering, classification, and embedding. In *Advances in Neural Information Processing Systems*, 2006.

[27] X. Zhu. Semi-supervised learning literature survey. *Computer Science, University of Wisconsin-Madison*, 2:3, 2006.