Performance evaluation of recommendation algorithms on Internet of Things services

Ibrahim Mashal, Osama Alsaryrah *, Tein-Yaw Chung

Innovation Center for Big Data and Digital Convergence, Department of Computer Science and Engineering, Yuan Ze University, Taoyuan, Taiwan

A R T I C L E   I N F O

Article history:
Received 28 October 2015
Available online xxxx

Keywords:
Internet of Things
Service recommendation
Tripartite graph
Collaborative filtering
Hyper-edge

A B S T R A C T

Internet of Things (IoT) is the next wave of industry revolution that will initiate many services, such as personal health care and green energy monitoring, which people may subscribe for their convenience. Recommending IoT services to users based on objects they own will become very crucial for the success of IoT. In this work, we introduce the concept of service recommender systems in IoT by a formal model. As a first attempt in this direction, we have proposed a hyper-graph model for IoT recommender system in which each hyper-edge connects users, objects, and services. Next, we studied the usefulness of traditional recommendation schemes and their hybrid approaches on IoT service recommendation (IoT-SRS) based on existing well known metrics. The preliminary results show that existing approaches perform reasonably well but further extension is required for IoT-SRS. Several challenges were discussed to point out the direction of future development in IoT-SR.

© 2016 Published by Elsevier B.V.

1. Introduction

Technological revolution in communication and embedded computing has led to new paradigm called Internet of Things (IoT) [1]. IoT attempts to connect uniquely identified and addressed objects to Internet based on standard communication protocols. Examples of things include smartphones, power meters, heart beat monitors, temperature meters, and various sensors that can be equipped with processor and memory to become smart objects. Analysts predict that the number of interconnected objects will reach 212 billion by 2020 [2].

Nowadays, third party service providers have offered many IoT based innovative and valuable services. For instance, Telus has launched its IoT marketplace in December 2014 featuring 75 different services and solutions hoping that the number will grow to more than 100 by the end of this year [3]. Telus services include: fleet management solutions, oil and gas solutions, retail solutions, restaurant solutions, construction solutions, and public safety solutions. Another example of IoT marketplace is Libelium [4] which has listed 54 IoT applications grouped by 12 vertical markets including: urban and remote environments, agriculture and farming, water quality, security and emergencies, retail, logistics, domestic automation and e-health.

As time goes on, more smart objects along with countless services will be introduced and users will subscribe or own more of these value-added services. This causes high complexity on recommending appropriate services from increasing possible sets of service based on various smart objects owned by the users and their needs. To resolve this complex problem, Recommender Systems (RSs) is an effective solution [5,6].

* Corresponding author.
E-mail addresses: osanjii@yahoo.com (O. Alsaryrah), csdchung@saturn.yzu.edu.tw (T.-Y. Chung).

http://dx.doi.org/10.1016/j.physa.2016.01.051
0378-4371/© 2016 Published by Elsevier B.V.

Please cite this article in press as: I. Mashal, et al., Performance evaluation of recommendation algorithms on Internet of Things services, Physica A (2016), http://dx.doi.org/10.1016/j.physa.2016.01.051
RSs are software systems that analyze information about items, users, and interactions between them in order to recommend the most suitable items to users by predicting their interest in a particular item. RSs have demonstrated their effectiveness in different domains, especially in the e-commerce domain. However, it is not examined and fully studied for the IoT. This paper addresses the problem of recommending IoT third-party services to users and evaluates the usefulness of existing recommendation algorithms on IoT service recommendation. To achieve the goal, we first introduce a tripartite graph-based model for IoT systems and project it into three bipartite graphs by utilizing the ternary relations hyper-edges among users, objects, and services to recommend services to the users. The graph is built by exploring users’ ownership of objects, objects utilized by services, as well as user subscription to services. Second, we formalize the service recommendation problem based on graph approach and analyze the various entities and heterogeneous relationships to uncover correlations between objects, services, and users. Based on the graph model, we implement various existing recommendation algorithms and their combinations to study their behavior in IoT service recommendation. Finally, we evaluate their performance based on some well-known metrics such as recall, precision, etc. The results illustrate that existing schemes worked decently but further extension is required to meet the challenges of IoT service recommendation. To the best of our knowledge, our work is the first attempt to design a service recommender system in the IoT.

The rest of the paper is organized as follows. Section 2 introduced related work and recommender system preliminary. Section 3 depicted the general system architecture. Section 4 presented a motivating example to illustrate the urgent need for IoT SRS. Section 5 introduced the proposed tripartite graph-based model that employs information of services and ambient objects for services recommendation. Furthermore, we give formal representation of IoT and introduced the formal model of the IoT SRS algorithm. Section 6 described in detail our datasets and the evaluation metrics used to validate IoT SRS. Preliminary results and evaluations were given in Section 7. Finally, Section 8 concludes the paper.

2. Related work

Traditional RSs use different techniques including content-based filtering (CB), collaborative filtering (CF), and hybrid techniques. As its name suggests, CB requires textual information about the items and the historical records of users [7]. CB recommends items similar to the items that active users have previously consumed or liked. It is based on description of item characteristics and a profile of the user’s preference. However, it requires a mechanism to associate content to many heterogeneous networked objects. Moreover, only very similar items to previous items consumed by the users are recommended, which creates a problem of overspecialization [8].

In contrast to the CB filtering, CF does not rely on items representations and its content [9]. Instead, it relies on the opinions of other people who share similar interests. CF requires additional rating system to capture and store users’ ratings. CF generates recommendations by identifying users who have similar taste for items and recommend items that they have liked. It has been found that CF is a powerful technique that is able to produce high quality recommendations and thus it is widely used nowadays.

Fundamentally, CF can be classified into user-based and item-based approaches [10]. In the user-based CF, the K most similar users with similar rating are found, and then their ratings are used to calculate a prediction for active users. The item-based CF takes into account the similarity between items themselves [11]. There are many approaches to find similarity. The most popular similarity metrics are Pearson correlation and cosine similarity. However, constructing a collaborative user model is not a trivial task. Furthermore, CF requires an up-to-date dataset of users and their preference, which is difficult to gather specially for a large number of objects in the IoT.

To achieve higher performance and overcome the drawbacks of aforementioned techniques, hybrid technique has been proposed [12]. Hybrid RSs combine collaborative and content information by one of seven basic hybridization mechanisms by combining features of two or more recommendation techniques.

In recommender systems, the graph-based approach has been tested in the past in different domains and has shown promising results. For example, tag recommenders construct a graph with users, resources and tags, and recommend a set of tags for a given user based on previously used and assigned tags [13,14]. Many meaningful research works on tag recommendation have been proposed in recent years [15]. A well-known tag recommender approach is the FolkRank (FR) [16,17] which adapts the Google PageRank algorithm to rank the nodes within a graph based on their importance in the network. A different mechanism is based on the classic CF approach that has been adopted for tags recommendation in Refs. [18,19]. Collaborative approaches exploit the relations between users, resources and tags of the folksonomy graph to select the set of recommended tags. Jäckel et al. [20] evaluate and compare user-based collaborative filtering, graph-based, and counting co-occurrences algorithms for tag recommendation. Other studies focus on using different approaches such as Rendle et al. [21], Wetzker et al. [22], Lops et al. [23], and Rawashdeh et al. [24]. Rawashdeh et al. applied the Katz measure to weighted undirected tripartite graph to provide tag recommendations for individual users.

Research work on recommendation for the IoT is in its infancy. To the best of our knowledge, there are very few articles that discussed recommendation in the IoT environment and current works are very primitive. For example, the work in Refs. [25,26] addressed things recommendation in IoT. They propose a framework to recommend the right thing to use at the specific time by exploring users’ relations and things correlations. However, they do not consider recommending third party services. Compared with this research effort, we constructed the services–things correlation graph to capturing similarities between services.
3. Architecture of IoT-SRS

MUL-SWoT is a lightweight RESTful platform for developing IoT and Social Web of Things (SWoT) applications and services. MUL-SWoT establishes connection between users and their smart objects to allow smart objects to communicate with Social Networks Sites (SNS) [27]. The platform collects, stores, and analyzes data collected from users’ smart objects and then posts the data on users’ SNS. Thus, it allows users to share objects and services with their friends and people they know. MUL-SWoT provides a RESTful API to enable 3rd party service providers to get access to MUL-SWoT functions. The MUL-SWoT design includes a number of modules and sub-modules, as shown in Fig. 1.

The core component of the platform is the MUL-SWoT management module that manages all other components. MUL-SWoT management module is divided into a number of sub-modules: Communication manager sub-module handles all communications through the MUL-SWoT. Configuration registry sub-module manages and stores all configurations about services and objects. Device management sub-module is responsible for managing all objects registered on the MUL-SWoT. Control and monitor sub-module controls objects and services and monitors their actions based on stored configurations.

An important component of MUL-SWoT is the service management module which is responsible for managing and storing information about service. Two sub-modules are found in the service management, namely Third party service registry management and Third-party service recommender. Third party service registry management sub-module controls and manages third party accounts and all services registered on the MUL-SWoT. Moreover, it generates service ID for the third party service provider.

Third-party service recommender is a sub-module of the service management module. The aim of this module is to assess which are the best services and returns top-ranked services that best matching users’ needs based on things they own. The recommender runs a large number of recommendation algorithms and is implemented through a four-step process consists of filtering, scoring, ranking, and evaluating. Filtering excludes services that do not match users’ needs. Scoring assigns a numeric value to each service. Ranking returns an ordered list of services based on filtering and scoring results. Evaluating uses standard Information Retrieval (IR) metrics to evaluate recommended services and recommendation algorithms. Based on the result from evaluation component, the top three recommendation algorithms that have the highest performance are then combined and integrated in attempt to produce more accurate and useful recommendations.

4. Motivating example

Consider a restaurant and food producers scenario where the owner Bob wants to monitor food conditions in order to comply with the food safety regulations. Traditionally, Bob and the employees collect manually the readings about temperature in refrigerators, freezers, and ovens to decide whether food is safe or not. In order to improve the accuracy and reduce time, Bob installed a number of sensors to measure and report the refrigerator temperature, air temperature, food temperature, and humidity. However, this is not enough to satisfy his needs. Without a powerful service that can collect and store these readings to be analyzed, Bob cannot be sure of his business and that his restaurant reputation is protected. Bob needs a service to monitor food safety status and issue alerts if anything goes wrong (e.g., a cold storage failure). Moreover, Bob needs to monitor the food status and every sensed data from anywhere and anytime. Bob asks suggestions to our IoT-SRS about possible services that can make use of the objects he owns. IoT-SRS recommends him a SafeFood service offered by blueRover Inc, which is best to meet his needs.

Please cite this article in press as: I. Mashal et al., Performance evaluation of recommendation algorithms on Internet of Things services, Physica A (2016), http://dx.doi.org/10.1016/j.physa.2016.01.051
5. Proposed approach

IoTSRS must draw users’ attentions to new services they do not subscribe yet. These services must be suitable for the objects owned by the users. Intuitively, objects may be used by more than one service, which mean that the service that shares some objects should not be excluded from recommendation. In order to develop IoTSRS the first step is to create a model of the IoT. In this section, we formalized the notion of IoT systems then we introduced and formulated several common recommendation techniques which are to be evaluated in IoTSRS.

5.1. A formal model for IoTSRS and problem definition

In the IoT, the relation between users, objects, and services can be modeled as a tripartite graph with hyper-edges between them as shown in Fig. 2. A tripartite graph is a graph with its vertices partitioned into three disjoint sets: $U = \{U_1, U_2, \ldots, U_m\}$ denote a set of $m$ users, $O = \{O_1, O_2, \ldots, O_n\}$ denote a set of $n$ objects, and $S = \{S_1, S_2, \ldots, S_k\}$ denote a set of $k$ services. We define $Y$ as a ternary relation between these three components that represent users subscription of services based on the objects they owns. $Y$ is defined in (1).

$$Y \subseteq \{u, o, s : u \in U, o \in O, s \in S\}.$$

Thus, the IoT system can be defined as a tuple that describes the users $U$, services $S$, objects $O$, and the ternary relation between them. The tuple is given in (2).

$$I = (U; S; O; Y).$$

The hyper-graph can be projected into three bipartite graphs that represent relations between objects–services ($OS$), users–services ($US$), and users–objects ($UO$). The relation between objects and services ($OS$), given in (3), uses weights to represent the number of users who subscribe service with their objects.

$$OS (o, s) = |\{y = u, o, s \in Y : u \in U\}|.$$

The relation between users and services ($US$) given in (4), where the weights represent the number of objects that are required to compose a service.

$$US (u, s) = |\{y = u, o, s \in Y : o \in O\}|.$$

The relation between users and objects ($UO$) is given in (5), represents the number of services sharing the same objects for users.

$$UO (u, o) = |\{y = u, o, s \in Y : s \in S\}|.$$

5.2. Internet of things service recommender systems (IoTSRS)

The task of IoTSRS is to rank all services provided by the 3rd party, filter services, and recommend a set of services $s \in S$ for a given user $u \in U$ and a given object $o \in O$. It takes a set of users, a set of objects, and a set of services as input and outputs...
a list of ranked services. IoTSRS implements several existing recommendation algorithms developed for recommendation using a tripartite graph. In this subsection we review these techniques.

1. Most-Popular-Service (MPS)
   This approach is the most basic approach that recommends most popular services to users. For any user \( u \in U \) and any object \( o \in O \) the same set of services \( S(u, o) \) is recommended. This set of services is weighted by count frequency in all service subscription \( Y_s \). MPS is given in (6).
   \[
   S(u, o) = \operatorname{argmax}_{s \in S} (|Y_s|).
   \]  

2. Most-Popular-Service-User (MPSU):
   It suggests the most frequent service within the services user subscribed regardless of the objects. It is personalized extension of MPS. We defined MPSU as in (7).
   \[
   S(u, o) = \operatorname{argmax}_{s \in S} (|Y_{s,u}|).
   \]  

3. Most-Popular-Service-Object (MPSO):
   MPSA suggests services that are most frequent with a particular object regardless of the users. We defined MPSO as in (8).
   \[
   S(u, o) = \operatorname{argmax}_{s \in S} (|Y_{s,o}|).
   \]  

4. Most-Popular-Service-User-Object (MPSUO):
   This algorithm mixes the Most-Popular-Service-User with the Most-Popular-Service-Object. MPSUO is given in (9).
   \[
   S(u, o) = \operatorname{argmax}_{s \in S} \left( \beta |Y_{s,u}| + (1 - \beta) |Y_{s,o}| \right)
   \]  

where \( \beta \) is used to balance the influence of both components; the user and the object components.

5. Servrank (SR):
   Servrank is inspired by the famous algorithm called FolkRank. ServRank computes preference vector from the tripartite graph \( G = (V, E) \), where \( V = U \cup S \cup O \) is the set of all vertices in the graph, which is composed of users, objects, and services. \( E \) is the set of the edges in the graph, which is defined by the three projections of the hyper-graph \( OS, US, \) and \( US \) presented in the previous sub-section. ServRank vector is given by (10).
   \[
   w = dA w + (1 - d) p
   \]  

where \( A \) is the normalized column-stochastic version of the adjacency matrix \( w \) of graph \( G \), \( p \) is the preference vector, and the dampening factor \( 0 < d \leq 1 \) determines the influence of \( p \). The ServRank vector is taken as a difference between two computations: one with a preference vector and one without the preference vector. The ServRank vector is defined in (11).
   \[
   w = w_1 - w_0.
   \]  

6. User-Based Collaborative Filtering (UBCF)
   IoTSRS also implements the commonly used algorithm CF. However, traditional CF cannot be applied directly and must be modified to cope with the ternary relations in the IoT tripartite graph. CF is based on finding similarity between users \( u \) and \( v \) using different similarity matrix. In IoTSRS, user \( u \) is modeled as vector over set of services where the weight is the projection \( US (u, s) \) and neighbors \( N^k_u \) of a user \( u \) are formed based on the set of services in the user profile \( Y_u \) and only the subset \( Y_s \) of users that have subscribed a service for active object \( o \) are taken into account when calculating the user neighborhood. Several techniques are used to calculate similarity. In IoTSRS, Jaccard’s similarity is used. The set of \( n \) recommended services can then be determined based on this neighborhood as given in (12).
   \[
   S(u, o) = \operatorname{argmax}_{s \in S} \left( \sum_{s \in N^k_u} \operatorname{sim}(Y_u, Y_s) \ast \delta (o, s) \right)
   \]  

where \( \delta (u, o, s) = \begin{cases} 1 & \text{if } (u, o, s) \in Y_s \\ 0 & \text{else.} \end{cases} \)

UBCF scales well with very large datasets since UBCF considers only those users that have subscribed same services with similar objects and thus the number of similarities to calculate is drastically reduced and thus reducing computation burden. However, the algorithm may not be able to recommend some services since user’s neighbors do not subscribe these services.

Please cite this article in press as: I. Mashal, et al., Performance evaluation of recommendation algorithms on Internet of Things services, Physica A (2016), http://dx.doi.org/10.1016/j.physa.2016.01.051
7. Object-Based Collaborative Filtering (OBCF)

In IoT-SRS, object \( o \) is modeled as vector over set of services where the weight is the projection \( OS (o, s) \). When a user selects an object to subscribe a service, the cosine similarity between this object and every object is calculated and neighbors \( N^s_o \) are then constructed. We defined OBCF in (13).

\[
S (u, o) = \arg \max_{s \in S} \left( \sum_{j \in N^s_o} \sim(Y_i, Y_j) \ast \delta (u, j, s) \right)
\]

(13)

where \( \delta (u, j, s) = \begin{cases} 
1 & \text{if } (u, j, s) \in Y_s \\
0 & \text{else.} 
\end{cases} \)

8. Hybrid Service Recommendation (HSR)

Some of the aforementioned individual algorithms do not perform well when they run alone. Combining different individual algorithms together will produce an effective and enhanced composite recommender that outperforms its constituent parts. IoT-SRS combines several recommenders together to produce hybrid recommenders.

A. Dual Hybrid Service Recommendation (DHSR)

In DHSR, the top three individual algorithms are combined together differently to get three DHSR. We have selected a linear model which combines pairs of recommenders on weighted hybrid recommenders where each model is trained separately. The final result of DHSR is the normalized result of each recommendation approach on the scale of 1. The combined linear model is given in (14).

\[
S (u, o) = \arg \max_{s \in S} (\beta \ast S_o (u, o) + \alpha \ast S_b (u, o))
\]

(14)

where \( \beta \) and \( \alpha \) are used to control the contribution of the two recommenders where \( \beta = 1 - \alpha \). Intuitively, when \( \alpha \) is set to 0, recommender \( a \) acts alone and recommender \( b \) will have no effect on the final result. In the case that \( \alpha \) is set to 0.5, each recommender contributes equally to the final result.

B. Triple Hybrid Service Recommendation (THSR)

THSR combines the top three algorithm together into one algorithm by using weighted linear combination as given in (15).

\[
S (u, o) = \arg \max_{s \in S} (\alpha \ast S_o (u, o) + \beta \ast S_b (u, o) + \gamma \ast S_c (u, o))
\]

(15)

such that weights \( \alpha + \beta + \gamma = 1 \) and all values are positive. If \( \alpha \) is set near 1 then hybrid would rely mostly on recommender \( a \) and other recommenders will have no effect on the final result.

6. Experimental setup

In this section we describe in detail the datasets, the evaluation method and the metrics used to validate IoT-SRS. The code we used for IoT-SRS implementation is based on the open-source code provided by TagRec [28].

6.1. Datasets

To perform reliable experiments, it is ideal to use a large scale datasets which are important for effective evaluation of recommendation algorithms. Unfortunately, there is a lack of publicly accessible well-known large scale data base about objects, IoT services, and user subscription in IoT services. The acquisition of datasets for IoT-SRS for testing algorithms is a challenge. Therefore, we used real-world data to evaluate the recommendation algorithm. Service and object profiles are created to comprise 110 objects, 90 services collected from Libelium, Telus, and blueRover catalogs. A dataset was collected from 400 individual users. Each participant was asked to select a number of objects he owns, or wishes to own from an object list. In addition, the users were asked to arbitrarily select IoT services they are interested in from the default list. We followed a standard procedure in recommender research and divided the dataset as follows: 80% for training set which is used to generate recommendation lists and 20% for the test set which is used to verify the quality of the recommendations.

6.2. Evaluation method

To evaluate prediction quality and performance of different algorithms implemented in IoT-SRS, we compared the top recommended services using a set of various well-known evaluation metrics as follows.

1. Recall (R): is a metric for completeness of recommendation result. Recall is calculated as the number of correctly recommended services divided by the number of relevant services. Recall is defined in (16).

\[
R@K = \frac{1}{|U|} \sum_{u \in U} \left( \frac{|s^k_u \cap S_u|}{|S_u|} \right)
\]

(16)

where \( s^k_u \) denotes the top \( k \) recommended services and \( S_u \) the list of relevant services of user \( u \in U \).
2. Precision (P): is a metric for exactness of the recommendation results that is calculated as the number of correctly recommended services divided by the number of recommended services. Precision is defined in (17).

\[
P@K = \frac{1}{|U|} \sum_{u \in U} \left( \frac{|S_u^k \cap S_u|}{|S_u^k|} \right). \tag{17}
\]

3. F-measure: is another metric to compare the performance of algorithms. It is a measure of a test’s accuracy that combines precision and recall into one score. F-measure is defined in (18).

\[
F - \text{measure} = \frac{1}{|U|} \sum_{u \in U} \left( 2 \times \frac{P@K \times R@K}{P@K + R@K} \right). \tag{18}
\]

4. Mean Reciprocal Rank (MRR): is the sum of the reciprocal ranks of all relevant services in the list of the recommended services. This means that a higher MRR is achieved if the relevant services occur at the beginning of the recommended list. MRR is defined in (19).

\[
MRR = \frac{1}{|U|} \sum_{u=1}^{|U|} \left( \frac{1}{|S_u|} \sum_{s \in S_u} \frac{1}{\text{rank}(s)} \right). \tag{19}
\]

5. Mean Average Precision (MAP): is an extension of the precision metric that also looks on the ranking of the recommended services. MAP is defined in (20).

\[
\text{MAP} = \frac{1}{|U|} \sum_{u=1}^{|U|} \left( \frac{1}{|S_u|} \sum_{k=1}^{K} B_k \times P@K \right) \tag{20}
\]

where \(B_k\) is 1 if the recommended service at position \(k\) is relevant.

6. Discounted Cumulative Gain (DCG): In DCG, the gain of a recommended item is discounted logarithmically with respect to its position in the recommendation list [29]. The DCG of a list of \(k\) services is defined as in (21).

\[
\text{DCG@K}(u) = \sum_{j=1}^{K} \frac{g_{u,s(j)}}{\text{max}(1, \log_b(j))} \tag{21}
\]

where \(g_{u,s(j)}\) is the gain of user \(u\) when service \(s\) is recommended. \((j)\) denotes the \(j\)th service in the recommendations ordered list, and the logarithmic base \(b\) is suggested to be 2 to ensure all positions are discounted. A normalized version of DCG, called NDCG is given in (22).

\[
\text{NDCG@K}(u) = \frac{\text{DCG@K}(u)}{\text{DCG@K}^*(u)} \tag{22}
\]

where \(\text{DCG@K}^*(u)\) is the ideal \(\text{DCG@K}(u)\).

7. Preliminary results

In order to validate the efficiency and evaluate existing algorithms used in IoTSSRS, we run a number of experiments. In the experiment, several variables are tuned. First, the neighborhood number \(K\) in UBCF and OBCF was tuned in increments of one from zero. The best result was reported when \(K\) is equal to 20. When \(K\) increases beyond 20, performance suffers from diminishing returns. Second, for the SR algorithm the parameter \(d\) was set to 0.7 and the number of iterations was set to 10. Finally, \(\beta\) was set to 0.5 in MPSO.

7.1. Individual algorithm performance evaluation

The preliminary results of our evaluation of the dataset are based on Precision/Recall plots. Fig. 3 reports the Precision/Recall of the individual recommendation algorithms. Comparing individual recommendation algorithms, the SR is the strongest individual recommender with the highest level of Precision/Recall estimates. The second most successful recommender algorithm is MPSO, meaning that frequency-based algorithms perform better when they consider objects. MPSO is able to provide relevant recommendations specially when there is no much information available about users and services they subscribed. MPSO outperforms all other frequency-based algorithms. OBCF comes third of the individual recommender algorithms. OBCF performs far better than UBCF, which reveals that services are better modeled by objects than by users. Other algorithms perform moderately well. However, all the recommender algorithms perform far better than the un-personalized MPS algorithm which simply employs the popularity of a service.
IoT has special characteristics and crucial features that make some applications suffer from some shortcomings and add more complexity to IoT application and services. For example, IoT is a highly heterogeneous network that deploys a large number of different types of objects and produces a large number of different services. Thus, IoT SRS must analyze the heterogeneous relationships between different users, objects, and services. The tripartite graph is a powerful tool that is capable to capture, identify, and comprehensively analyze the complex relations between nodes in IoT systems. Based on this discussion, it is intuitive that SR, which is a graph-based recommendation algorithm, achieves more accurate predictions and more reasonable recommendation than other algorithms (such as CF or the basic graph-based algorithm MPS).

The SR algorithm in IoT SRS also handles another critical feature of IoT system. IoT produces a vast amount of data and many services from a huge number of objects. The SR algorithm in IoT SRS can effectively mitigate the sparsity problem by creating more paths between users, objects, and services nodes. The recommendation algorithm based on the graph model has an advantage on solving the sparsity problem with both high accuracy and little computation time. Another reason for the superiority of SR is its independency of contextual information. In IoT, the resource-constrained nature imposes many constraints that make contextual information hard to specify, even with the use of semantic technologies. Moreover, IoT services descriptions are usually incomplete and ambiguous.

On the other hand, although the un-personalized approach MPS is the simplest approach and its performance is the worst as seen in Fig. 3. It still can be applied in IoT SRS since it requires little online computation and suffers minimal from cold-start problems. Thus, at the initial stage of IoT SRS, MPS could be an option.

Table 1 shows a comparison of results for MRR, MAP, and NDCG for both K = 5 and 10. The results clearly show that both SR and MPSO algorithms outperform all other individual algorithms.

### Table 1
Comparison of results for MRR, MAP, and NDCG.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Measure</th>
<th>MRR@5</th>
<th>MAP@5</th>
<th>NDCG@5</th>
<th>MRR@10</th>
<th>MAP@10</th>
<th>NDCG@10</th>
</tr>
</thead>
<tbody>
<tr>
<td>MPS</td>
<td></td>
<td>0.086788</td>
<td>0.09399</td>
<td>0.141666</td>
<td>0.102341</td>
<td>0.118519</td>
<td>0.193908</td>
</tr>
<tr>
<td>MPSU</td>
<td></td>
<td>0.230302</td>
<td>0.261417</td>
<td>0.335579</td>
<td>0.248704</td>
<td>0.300707</td>
<td>0.394514</td>
</tr>
<tr>
<td>UBCF</td>
<td></td>
<td>0.267604</td>
<td>0.294258</td>
<td>0.353345</td>
<td>0.271717</td>
<td>0.305076</td>
<td>0.367613</td>
</tr>
<tr>
<td>MPSUO</td>
<td></td>
<td>0.504488</td>
<td>0.570987</td>
<td>0.653029</td>
<td>0.528267</td>
<td>0.631598</td>
<td>0.734624</td>
</tr>
<tr>
<td>OBCF</td>
<td></td>
<td>0.533274</td>
<td>0.63664</td>
<td>0.719554</td>
<td>0.551486</td>
<td>0.692901</td>
<td>0.785377</td>
</tr>
<tr>
<td>MPSO</td>
<td></td>
<td>0.569914</td>
<td>0.702872</td>
<td>0.780856</td>
<td>0.582504</td>
<td>0.749283</td>
<td>0.829517</td>
</tr>
<tr>
<td>SR</td>
<td></td>
<td>0.574332</td>
<td>0.704967</td>
<td>0.789098</td>
<td>0.58843</td>
<td>0.75125</td>
<td>0.832668</td>
</tr>
</tbody>
</table>

### 7.2 DHSR performance evaluation

DHSR combines the best three individual recommender algorithms earlier discovered. SR, MPSO, and OBCF are well represented in the DHSR algorithms. In Fig. 4, we see that the three DHSR approaches dramatically surpasses the individual components. We evaluated the weights $\alpha$ and $\beta$ in 0.05 increments to consider every possible combination. For OBCF + MPSO DHSR best results were found when $\alpha = 0.20$ and $\beta = 0.80$, meaning that MPSO contributes 80% of the DHSR and OBCF contributes 20%. Best results for SR + OBCF were found when $\alpha = 0.85$ and $\beta = 0.15$, meaning that SR contributes 85% of the hybrid model and OBCF contributes 15%. When looking at their Precision/recall, OBCF + MPSO and SR + OBCF achieve nearly identical results. This is due the fact that OBCF is the worst algorithm among these three. Although OBCF contribution is limited in the DHSR, but it degrades the overall result of DHSR, SR + MPSO achieve the best performance of the DHSR.
when $\alpha$ and $\beta$ were set to 70% and 30% for SR and MPSO, respectively. SR + MPSO adds to the strength of SR the strength of the frequency-based algorithms which is based on frequency counting and thus achieve higher score.

7.3. THSR performance evaluation

In the THSR, the three recommendation algorithms MPSO, OBCF, and SR are combined to contribute to the hybrid. MPSO + OBCF + SR perform very well as one might have expected and outperforms the individual contributions of its components and outperforms the DHSR algorithms. This situation was reached when $\alpha = 0.10$, $\beta = 0.40$, and $\gamma = 0.50$. As such SR contribution to the hybrid was 50%, MPSO accounts for 40% of the model, and the remaining contribution is from OBCF. Fig. 5 shows the performance of the THSR.

As shown in this figure, the THSR algorithm shows a clear edge over the DHSR algorithms on precision and recall. More specific, the THSR achieves higher recall with more than 13.5% compared to OBCF + MPSO, 8.6% compared to SR + MPSO, and more than 9.5% compared to SR + OBCF. Comparing precision, the precision of the THSR rises by more than 0.2%, 0.52%, and 0.4% compared to SR + MPSO, OBCF + MPSO, and SR + OBCF, respectively. Such improvements prove the effectiveness of the THSR over the DHSR and individual recommendations approaches.

Apparently the precision and recall would be different with different number of recommended services. Fig. 6 presents the relation between both precision and recall against the number of recommended services. To illustrate the influence of the number of recommended services on recommendation schemes, we take the recall as an example. For the THSR, when the service number changes from two to ten, the recall increases about 1.6 times. While for the DHSR it increases by 1.8, 1.7, and 1.6 times for OBCF + MPSO, SR + OBCF, and SR + MPSO, respectively. Such difference implies that THSR has a higher completeness than DHSR.

From the experimental results and its analysis we can see that hybrid recommendation schemes, both DHSR and THSR, not only outperform the frequency-based baseline recommenders but also outperform more effective graph-based algorithms such SR. Moreover the hybrid scheme is computationally efficient and thus is suitable for deployment in large networks such the IoT.
However, there still are some important IoT characteristics which have not been addressed in existing algorithms. For example, IoT infrastructure is deployed in many geographically distributed places and the services are Location-Based Services (LBS). None of existing algorithms consider this characteristic. In other words, users may be surrounded by various objects, in addition to their own objects, when they stay in different area. It indicates that a service can be enriched in power when more objects join. It also means that a service must be flexible and be agile in environment. Thus, new attributes must be considered when recommending location-aware IoT services. Mobility is another example. An important source of complexity in IoT is mobility because objects are organized in changing structures and thus objects may not stay at an ambient area all the time. When recommending a service, such features must be captured by the graph model and existing algorithms must be extended to cope with mobility problem.

8. Conclusion

Recommendation of services in IoT is crucial to facilitate popularity of IoT. In this paper we examined the possibilities of leveraging existing recommendation algorithms, specially graph-based, in IoT. We have presented a graph based model for IoT systems and showed how existing graph-based recommendation algorithm can be extended to recommend services in the IoT. The results of our experiments not only show that the schemes used for the tripartite graph recommendation perform reasonable and some hybrid schemes produce high quality results. However, current graph model for IoT service recommendation is very primitive. It cannot capture localization of sensor interaction and mobility of sensors in IoT services. Thus, existing model must be extended to specify the new features and new schemes must be developed.

Acknowledgment

This work was supported by the Ministry of Science and Technology of Republic of China, Taiwan, under contract number MOST XXX-XXXX-X-XXX-XX.

References


