A Decentralized Recommendation Engine in the Social Internet of Things

Daniel Defiebre
UX, Digital Supply Chain and Manufacturing
SAP SE
Walldorf, Germany
daniel.defiebre@aikq.de

Dimitris Sacharidis
E-Commerce Research Unit
TU Wien
Vienna, Austria
dimitris@ec.tuwien.ac.at

Panagiotis Germanakos
UX S/4HANA, Product Engineering, IEG
SAP SE
Walldorf, Germany
panagiotis.germanakos@sap.com

ABSTRACT

In the Social Internet of Things (SIoT), the connected objects operate autonomously to request and provide information and services to end users. Following concepts and aspects from human social networks, the objects interact with each other, and over time develop trustworthy relationships. By mitigating security and privacy concerns, the benefit to end users is more effective and trustworthy services. In this work, we design a recommender system over SIoT. The recommender takes advantage of the social dynamics that drive the behavior and interactions of autonomous objects as they attempt to discover and return the best possible result. The main aim is to facilitate the optimum pairing of objects so as to enable effective recommendations.

CCS CONCEPTS

• **Information systems → Recommender systems**.

KEYWORDS

Decentralized Recommender System; Internet of Things

ACM Reference Format:
Daniel Defiebre, Dimitris Sacharidis, and Panagiotis Germanakos. 2020. A Decentralized Recommendation Engine in the Social Internet of Things. In *Adjunct Proceedings of the 28th ACM Conference on User Modeling, Adaptation and Personalization (UMAP '20 Adjunct), July 14–17, 2020, Genoa, Italy*. ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3386392.3397602

1 INTRODUCTION

The Internet of Things (IoT) entails the interconnection of heterogeneous smart devices (or things) and services over a network infrastructure that facilitates their anytime and anywhere interaction and exchange of data [2, 15]. An overarching goal is that the interacting things will be able to maximize their outcome with minimal utilization of resources, by employing qualities like navigability, scalability, trustworthiness, and information availability, accessibility and shareability [4, 9]. Additionally, the vision is to be able to consider users' (owners) characteristics, preferences and wishes for providing more targeted and inclusive services on specific requests, although they interact autonomously in the space [3]. The latter consideration brings into the surface an enriched communication which takes place not only between the various things but also among people and between people and things. This reality refers to the Social Internet of Things (SIoT) which promises the alignment towards more human-centered workflows and seamless models of interaction between people and smart things within an intelligent social framework [9]. The expected benefits emphasize on the proactive discovery (*what*) and delivery of information and resources *when* and *how* the users need it most. The convergence of SIoT and personalized recommender technologies [27] can play a significant role towards that direction by taking advantage of the characteristics of the social networking models and interaction paradigms, and the sophistication of recommendation algorithms. They can help users to identify easier artifacts, saving time and effort, and improve their user experience. However, although recommendation techniques have been extensively applied in various domains like e-commerce, entertainment, news [23], creating a clear view of selection for the users (recommending songs, books, magazines, documents, etc.) by blurring out irrelevant for them information, their utilization in IoT, let alone SIoT, is still at early stages [22, 28]. Specifically, there is some relevant work discussing how to handle recommendations [24] or how to construct trust-aware social connections in the SIoT [19], but no concrete recommendation engine over SIoT has been proposed.

In this paper, we discuss a recommendation engine in SIoT that runs over a decentralized architecture. The main aim is to facilitate the optimum pairing of objects and the provision of best fit recommendations between them while interacting and exchanging in the dynamic virtual space. Key characteristics of the proposed recommender are: (1) The use of decentralized information over a fully dynamic social network (as opposed to static, that most recommenders currently run upon). (2) The distributed information among objects is not centrally controlled (i.e., information is not partitioned in a predefined way among objects). Only local information can be used when objects compute their recommendations — no object has access to the entire history of all objects. The object can decide to collect additional information from its ego-network, in order to improve the quality of the recommendations. (3) The objects decide themselves, based on a quality of friendship model, which connections to terminate so as to maintain the high quality of their neighborhood. This changes the social network and thus

UMAP '20 Adjunct, July 14–17, 2020, Genoa, Italy

ACM ISBN 978-1-4503-7950-2/20/07…$15.00
https://doi.org/10.1145/3386392.3397602

the way information is propagated (the recommender has no influence on the network formation), maintaining the dynamicity in the social network. (4) The objects communicate via the decentralized architecture with the goal of agreeing on a global model. The proposed recommender is one of the main components of a Dynamic & Anthropomorphic Network of Objects System (DANOS) detailed in [10, 11]. An evaluation using a small-scale dataset, finds that the SIoT recommender works effectively across the decentralized environment, and is able to make recommendations that challenge those of a recommender with complete knowledge of the data.

2 RELATED WORK

This paper presents a decentralized recommender that operates in the social internet of things. As such, it is related to approaches for decentralized recommendations, and IoT related recommenders.

Decentralized Recommenders. Decentralized recommenders use information that is distributed over a network of several nodes (peers). Each peer has only a partial, local, view of the information in the network, and is connected to other peers, from whom it may retrieve additional information. We classify decentralized recommenders according to the dynamicity of the network established.

First, we consider the case when the network is fixed, i.e., does not change or evolve over time. The works in this class essentially employ ideas from the literature of social-aware recommendations. For instance, [29] computes trust weights among peers, and then uses these weights in place of similarities in a user-based collaborative filtering (CF) technique, similar to [21]. Specifically, the type of information exchanged is a user profile (a history of ratings/feedbacks about some items), and a peer may request information from other peers up to some number of hops away. All collected profiles are weighted by the computed trust values and aggregated to predicted ratings/interest in unknown items.

In some other works, [5, 17] peers form connections via epidemic, or gossip-based communications with the goal of connecting to similar-minded peers. For this purpose, there is a similarity function involved, which typically computes rating/feedback similarity. After the network is formed, however, it is treated as fixed, i.e., the network will not change over time as recommendations are made and feedbacks are received. [17] employs a user-based CF techniques. The feedback from peers up to two hops away are collected (friends and friends-of-friends). Then a random-walk approach computes an adjusted similarity value between two peers, which is used to weigh the ratings. [5] employs a simpler recommendation engine, where peers simply send a list of recommended items, rather than their profiles.

Another approach is to connect peers with a distributed hash table (DHT) approach. In this case, the network is again fixed, but the DHT dictates how data is stored and connections are made. This means, that peers do not have control over their data, which raises privacy and security concerns. For instance, in [16], a peer uses a DHT to locate its most similar peers, retrieves their ratings, and then recommends using a plain user-based CF technique.

All aforementioned techniques are essentially a memory-based CF, where the information exchanged is the rating profiles. In contrast, in model-based CF techniques, like matrix factorization, the information exchanged is a local view of the model. However, for such an approach to work, each peer must store not only its own ratings, but also a part of the global rating matrix to be factorized [25]. Thus, privacy and security issues are also raised in this case.

There is another line of work, [6, 7, 26], where peers establish device-to-device (D2D), connections in an opportunistic manner, e.g., when they are in close proximity to each other. Therefore, the network continuously changes and the peers have no control over it. In this setting, all a peer can do is collect information, rating profiles, from the peers it has connected to it at some point in the past. Typically, user-based CF is used to make recommendations based on the collected profiles [6, 26]. [7] focuses on how to establish such D2D connections from a technical, networking standpoint, and is not concerned that much with recommendation techniques.

Compared to approaches in this category, our work differs in that it (1) considers a dynamic network of peers (the agents of objects acting on behalf of their owners) where the network evolves over time so as to improve recommendation effectiveness, and (2) provides increased security and privacy guarantees, as it restricts the information flow along the network.

IoT and Recommenders. Recommendations in the IoT are mostly treated as an orthogonal aspect to IoT. Refer to [12, 22] for a detailed overview of recommendation techniques suitable for IoT. Specifically, [12] discusses collaborative, content-based, utility-based, sequence-based, constraint-based techniques that can be applied as is. Moreover, [12] also presents some novel adaptations of recommendations methods for IoT. In general, IoT is seen as the means to collect more data about the user, such as the context [1], so as to provide better domain-independent recommendations.

In IoT, it is often desirable to recommend services offered by other objects. [18] studies the case where a service is to be offered to a group of users. Thus, a group recommendation approach is proposed, which is in fact independent of the IoT setting. As another example of service recommendation, [20] employ random-walk techniques, like PageRank, over a tripartite graph defined by owner-object-service relationships.

There are some few works that specifically target the SIoT domain. [24] discusses ideas and challenges of developing a recommender in SIoT, but offers no concrete implementation. [13] presents a discovery mechanism, rather than a recommendation engine in SIoT. A fixed network of agents is assumed, where the agents exchange their data (items to discover) so that eventually nearby agents posses similar data. Exchanging data instead of adapting social connections poses security and privacy threats.

There is another line of work addressing neighborhood formation (sometimes called friend recommendation) in SIoT. [8] discusses a task independent approach, where feedback from previous transactions among objects is used to define a metric suggesting when a friendship is to be established or cancelled. This idea of object-to-object transaction feedback also appears in our recommendation engine: feedback from the user on an a specific item, affects the similarity between the user's object and other objects. Neighborhood formation in SIoT may also depend on trust relationships among objects, e.g., as defined in social-aware recommenders. [19] discusses how to compute and maintain trust using transitive relationship, and also how to infer trust for unseen tasks. In our

system, a similar idea is conveyed by the object-object similarity metrics, which evolve over time based on users' feedbacks.

In conclusion, we note that ours is the only work that considers domain-independent recommendations in SIoT, proposes a concrete solution characterized by a novel neighborhood formation mechanism, and evaluates it in an SIoT simulator.

3 RECOMMENDATIONS IN SIOT

We first present the general-purpose SIoT system DANOS, and then discuss how to implement a recommender engine in this system.

3.1 The DANOS System

DANOS is a system for SIoT objects that runs over a decentralized architecture, introduced in [10, 11]. In what follows we provide a short overview of DANOS. An *object* in DANOS acts as an agent on behalf of its *user*, and thus has an *object profile* that consists of information specific to the user (e.g., preferences, personality traits) and specific to the object (e.g., device characteristics). The object profile guides the behavior of the object in the DANOS system.

DANOS contains virtual *intent areas*, that facilitate specific information needs. For example, there is an area devoted to product recommendations. An area is organized into *cells*, which act as virtual rooms where objects visit to establish friendships. We note that DANOS only facilitates the friendship establishment process. For security and privacy reasons, it does not facilitate the exchange of information, which only happens directly between objects that are friends.

An object with a specific task, e.g., to receive product recommendations, may decide that it needs to establish new friendships to address this task. In that case, it enters DANOS and the *intent manager* directs them to the area that best matches their intent. Once in an area, the object then receives a *travel schedule* that consists of a list of cells to visit in order. This list is compiled by the area's *schedule manager*, who matches the object profile with the various *cell proxies*. A cell proxy is a succinct representation of the cell, compiled by aggregating the profiles of objects who have sent a positive *feedback* to the cell. Based on the cell proxy, the schedule manager identifies cells where an object might find other objects with relevant information for the task.

Once in a cell, the object makes a friendship request. The *relationship manager* within a cell then matches the profile of the requester to those of other objects that have recently visited the cell. Among them, the relationship manager selects the most similar objects to become friends with the requester. Objects decide on their own when to cancel friendships.

3.2 A Recommendation Engine in DANOS

In this section, we describe how a decentralized recommendation system can be implemented within DANOS, make explicit the various aspects of DANOS we previously overviewed.

Preliminaries. In the following, we refer to a user and its agent interchangeably; we use the former in the context of recommendations, and the latter in the context of the SIoT network. A user u has a *profile* abstractly represented by the vector p_u. The profile contains information specific to the user (e.g., preferences, personality traits), and specific to the object (e.g., device characteristics).

We assume a specific intent for the SIoT objects: recommendations of *items* from a set I. We assume that an item $i \in I$ has a *content* (e.g., product categories) that is abstractly represented by the vector c_i. For example, the k-th dimension $c_i[k]$ of the content vector may indicate the membership degree of item i to the k-th product category.

A user u provides a *rating* to an item i, which is represented as $r_{u,i}$ and is normalized in the $[0, 1]$ range. We use I_u to denote the subset of items that user u has rated. The feedback given by a user determines the user's *preference* on item content, represented by the vector π_u. Specifically, the k-th dimension of the preference vector indicates the inclination of the user towards the k-the content aspect, and is thus computed as:

$$\pi_u[k] = \sum_{i \in I_u} c_i[k] \cdot r_{u,i}.$$

Given two users u, v, we define three similarity values, based on their profiles, their ratings, and their preferences.

The *profile similarity* between two users u, v is denoted as $s_p(u, v)$ and is computed as the cosine similarity of their profile vectors:

$$s_p(u, v) = \frac{\langle p_u, p_v \rangle}{\|p_u\| \|p_v\|}.$$

Similarly, their *preference similarity* is the cosine similarity of their preference vectors:

$$s_\pi(u, v) = \frac{\langle \pi_u, \pi_v \rangle}{\|\pi_u\| \|\pi_v\|}.$$

The *rating similarity* between two users u, v is denoted as $s_r(u, v)$ and is computed as the adjusted Pearson Correlation Coefficient of the users' ratings normalized to $[0, 1]$:

$$s_r(u, v) = \frac{1}{2}\left(\frac{\sum_{i \in I_u \cap I_v} (r_{u,i} - \overline{r}_u)(r_{v,i} - \overline{r}_v)}{\sqrt{\sum_{i \in I_u} (r_{u,i} - \overline{r}_u)^2} \sqrt{\sum_{i \in I_v} (r_{v,i} - \overline{r}_v)^2}} + 1 \right),$$

where $\overline{r}_u$ is the mean rating of user u.

Friendship Network. In SIoT, an object can only exchange information directly with other objects it has established a friendship with. In the context of recommendations, the information exchanged concerns the items, i.e., item content and ratings. Therefore, an object has only a limited view on the catalog I of items: it only knows what has been shared from its friends. Consequently, to be able to make good recommendations, an object needs to establish and maintain meaningful friendships.

At any point in time, a user u has a set of friends denoted as F_u. The information available to u consists of the content and ratings of all items rated by a friend.[1] Based on this information, user u computes the preference $s_\pi(u, v)$ and rating $s_r(u, v)$ similarity to each friend v. Moreover, when u established the connection with v in DANOS, u received the profile similarity $s_p(u, v)$ from the relationship manager — the profile vector p_v is considered private and is not shared.

So, for each of its friends, user u knows its preference, rating, and profile similarity. These similarity values are aggregated into an non-symmetric overall similarity. Specifically, the *overall similarity* of v to user u in the context of u's network is the convex

[1] In practice, the information is restricted to the most recent few ratings.

combination of the three similarities normalized among all friends, and is computed as:

$$s(u, v; F_u) = w_p \cdot s_p(u, v; F_u) + w_r \cdot s_r(u, v; F_u) + w_\pi \cdot s_\pi(u, v; F_u), \quad (1)$$

where the similarity weights w_p, w_r, w_π are in $[0, 1]$ and sum to 1; and each normalized similarity s_x for $x \in \{p, r, \pi\}$ is computed relative to the friends F_u of u as $s_x(u, v; F_u) = \frac{s_x(u,v)}{\sum_{v' \in F_u} s_x(u,v')}$. The similarity weights are external parameters whose values are determined empirically; in the simplest setting, all weights are equal.

For making recommendations, a user would want to be friends with users that are similar-minded, i.e., have high profile, rating, and preference similarity. However, having a highly similar circle of friends may cause a filter bubble, reducing the number of items the recommender may choose from (catalog coverage). Thus, to build a useful friendship network, the user should also consider the diversity of preferences among its friends.

We define the *preference diversity* of a friend v with respect to the friendship network of u as the average preference dissimilarity of v to every other friend of u:

$$d(v; F_u) = \frac{1}{|F_u \setminus \{v\}|} \sum_{v' \in F_u \setminus \{v\}} (1 - s_\pi(v, v')),$$

where v is excluded when computing the average. Intuitively, preference diversity quantifies how dissimilar a friend is with respect to the other friends of a user.

Combining the overall similarity of a friend v to user u and the preference diversity of friend v, we derive the *quality of friendship* (QoF) of friend v to user u. Similar to the MMR and xQuAD frameworks, we compute QoF as:

$$q(v; F_u) = (1 - \lambda) \cdot s(u, v; F_u) + \lambda \cdot d(v; F_u),$$

where the two terms are traded off by an external parameter λ. Intuitively, a friend v has a high QoF w.r.t. to u, if v is similar to u and/or if v's preferences are dissimilar to those of u' other friends.

Making Recommendations. When a user initiates a recommendation request, the objects performs the following tasks. First, it looks whether it needs to cancel friendships. Second, it may visit DANOS to establish new connections. Third, it makes a recommendation. Fourth, it provides cell feedback to DANOS. In what follows, we discuss the third task, and present the others later.

In DANOS, an object only uses information from itself and its friends to process a recommendation request. Specifically, we assume that the object has collected the content and ratings of all items rated by each of its friends. The recommender thus processes this information to derive its recommendations. Any collaborative filtering or content-based approach, or some combination thereof, can be used. For our purposes, we implement a simple hybrid user-neighborhood collaborative filtering method that also takes into account item content similarity. Specifically, we predict the rating of each item $i \in \cup_{v \in F_u} I_v$ known to the user u as:

$$\hat{r}_{u,i} = \overline{r}_u + \sum_{v \in F_u} s(u, v; F_u) \cdot (r_{v,i} - \overline{r}_v), \quad (2)$$

since the similarities are normalized, i.e., $\sum_{v \in F_u} s(u, v; F_u) = 1$.

Cancelling a Friendship. As users' preferences and rating history changes over time, so does the quality of friendship. A user should try to maintain a circle of friends that have high quality. Therefore, it may decide to cancel some friendship connections. We employ a simple outlier detection mechanism to cancel a friendship in the network of u. Consider the distribution of QoF values $q(\cdot; F_u)$ among the friends F_u. Let Q_ϕ denote the $1/\phi$-quantile of this distribution; for example when $\phi = 0.25$, we get the first quartile. Then define the difference $\Delta = Q_{1-\phi} - Q_\phi$; again when $\phi = 0.25$, this is the interquartile range. Then, we cancel friendship with any friend that has a QoF below $Q_\phi - \Delta$, i.e., is a negative outlier.

Establishing a Friendship. Each object has a target number of friends. When its network becomes smaller, the object might visit DANOS to establish more connections. Specifically, if the object is missing a ratio p of the target number of friends, the object chooses to visit DANOS with probability p. When it decides to visit DANOS, the object gets a travel schedule to visit cells. At each cell it visits, the object receives from the relationship manager, a number of objects that match the requester object. Recall that DANOS only knows the object profiles of users. Therefore, the relationship manager computes the profile similarities of the requester with every other object that has recently visited the cell. Then, the manager selects a number of other objects as friend suggestions to the requester. The requester then establishes these connections. If the requester object has more friend spots to fill, it proceeds to visit the next cell in its travel schedule. Otherwise, it exits DANOS.

Cell Proxy and Feedback. Recall that each cell in DANOS serves as a "meeting place" for SIoT objects. The effectiveness of the distributed SIoT environment depends on how well an object can identify like-minded objects. This implies that DANOS cells must become specialized enough with a purpose, e.g., this cell is frequented by objects that tend to like a specific class of products. To support this specialization, a cell must receive feedback on whether the object connections established within it were meaningful for the objects involved or not.

DANOS achieves this objective via two mechanisms: *cell proxy* and *cell feedback*. After processing a recommendation request, an object receives some user feedback. This user feedback changes the preference and rating similarity of the object to its friends, and ultimately the QoF with each friend. Some friends may become more useful (higher QoF) while others less useful (lower QoF) as a result. The object identifies the most useful of its friends, with a process similar to finding QoF outliers as discussed for friendship cancelling — only this time positive outliers (with QoF above $Q_\phi - \Delta$) are selected. For each such useful friend, the object creates a *cell feedback*, which merely consists of the object id of itself and its friend, and sends this information to the cell where the object connected with this particular friend.

Each cell maintains a fixed-size buffer of object profiles for those object ids found in recent cell feedbacks. Specifically, for the most recent cell feedback received, the two associated object profiles are inserted at the top of the buffer, while the bottom-two, oldest, object profiles are evicted. After processing a cell feedback, the cell creates a new *cell proxy*. The cell proxy is the average of all object profiles in this buffer. Recall that in DANOS, the cell proxy is used to determine a travel schedule for an object. Specifically, the travel manager computes the profile similarity $s_p(u, c)$ of the object to each cell proxy c, and compiles as travel schedule the list of cells ranked from most to least similar.

4 EVALUATION

Dataset. For the evaluation we use a dataset that contains ratings on topics (the items) from users. This dataset is compiled from a targeted small scale user study we conducted in January 2020, that asks professionals within a company for their profiles, and their ratings on a scale of 1–5 on the topics (i.e., technologies, tools, programs, applications, etc.). Profiles include predefined attributes used in the company for all employees, organized into 5 main categories: job description (with 5 attributes, e.g., current role, years of expertise); skills and experience (with 4 attributes, e.g., educational level, trainings, skills); learning preferences (with 4 attributes, e.g., preferred medium for learning, learning on-line or in classroom); about me (with 3 attributes, e.g., age, gender, topics of interest); personality (with 5 attributes, i.e., the personality traits of the Big Five Inventory — BFI) [14]. After distributing personal invitations to professionals that would fit the expected sample characteristics, we managed to have a consistent dataset containing 134 ratings from 48 professionals (11 female, 37 male) on 16 topics.

SIoT Simulation. Each user from the user study is associated with an SIoT object. Initially, each object inherits the profile of its user, and joins DANOS to make connections based on its profile similarity. After this bootstrapping phase, we randomly shuffle the available ratings, and scan them in sequence. For each rating, we create a recommendation request for the associate user. That is, the user's object needs to provide a recommendation and thus may choose to drop some friendships and enter DANOS to establish new connections. Regarding the DANOS configuration, we set the target number of friendships to 10. The number of cells in DANOS is varied among the values $\{1, 2, 4, 8\}$, and the corresponding methods are indicated as Danos-1, Danos-2, etc.

Baselines. We compare our recommendation strategy with two baselines. The first, called *Central*, is a centralized recommender system that has access to all data. The purpose of this baseline is to investigate whether the distributed environment and the profile specialization within cells can substitute for the missing global knowledge. The second baseline, called *Static*, is a restricted version of our SIoT recommender in that the social network is created once during the bootstrapping phase, and remains fixed ever since. Its purpose is to investigate whether the dynamic social network evolves effectively over time.

To keep the comparison fair, the baselines and our SIoT recommender use the same underlying recommendation strategy, the hybrid user neighborhood-based approach. The method only differ in what data they have access to. In Central, each user/object has access to all data. In Static and in our approach, an object can only access the data that its friends have, which essentially depends on the structure of the social network formed.

Evaluation Metrics. We measure the root mean squared error (*RMSE*) of the rating predictions made by the recommenders. We also count the ratio of requests for which the recommender does not return the item eventually ranked by the user, and denote this as *No-Rec*. Moreover, we use the positive ratings (with score 4 or 5) to measure the ranking accuracy of the recommenders in terms of the mean reciprocal rank (*MRR*). We report the mean metric values over three executions of the simulation.

Table 1: Recommendation Effectiveness

Method	No-Rec	RMSE	MRR
Central	61.7%	1.994	0.159
Static	60.2%	2.085	0.156
Danos-1	43.6%	1.936	0.199
Danos-2	44.4%	1.931	0.200
Danos-4	48.9%	1.915	0.203
Danos-8	48.1%	1.978	0.243

Results. Table 1 shows the evaluation metrics for the 4 configurations of DANOS and the 2 baselines. The reported values are means over all recommendation requests. Note that we prefer low values for No-Rec and RMSE and high values for MRR. In all metrics, the two baselines perform worse than the DANOS variants. The bigger differences are in terms of No-Rec, indicating that the baselines fail to recommend the ground truth item more than 60% of the time, whereas the DANOS approaches only fail around 40% of the time. This also implies that for the DANOS approaches, the RMSE is computed over more data points. In terms of MRR, there is also a significant difference, with the baselines exhibiting MRR around 0.15 and DANOS approaches around 0.20. When we compare the baselines, we find that Central performs slightly better, as it tries to optimize the user neighborhood at each request compared to Static.

At first glance, it may look surprising to find that Central, having knowledge of the entire data, performs worse than the DANOS approaches, having only partial knowledge of the data. The explanation is that Central uses its complete data knowledge to select the users to go in the neighborhood. At that point, it makes the best selection that it can. However, this optimal neighborhood formation does not necessarily mean more effective recommendations, as we actually observe in the experiments. Actually, the advantage in DANOS is that a user selects its friends from a *smaller pool* of users, those that have recently visited the cells. As a result, the pool of users is more *specialized* and thus better equipped to handle each recommendation request. The benefit of this specialization in DANOS is also apparent in that the performance improves, to some extent, as we increase the number of cells — making thus each cell even more specialized. For example, we see that the best RMSE occurs when we have 4 cells, while the best MRR occurs when we have even more, 8, cells. In conclusion, we observe that the *distributed nature* of DANOS helps achieve higher recommendation effectiveness that a centralized approach. Moreover, we also note that the *dynamicity of DANOS* is equally important for effectiveness — when restricted to a fixed network, as Static is, the effectiveness becomes worse than the centralized approach.

To better illustrate the recommendation effectiveness of DANOS over time, we compute rolling averages of RMSE and MRR over time; the results are shown in Figures 1 and 2, respectively. In general, we observe that both metrics improve over time, RMSE decreases, while MRR increases. At all time instances, Static performs the worst. It is worth observing that the DANOS approaches perform the best almost at all time instances. The centralized approach performs slightly worse than the DANOS methods in terms of RMSE, and slightly better than them in terms of MRR in the last few time instances. Among the DANOS methods, as before we note that Danos-4 is the best in terms of RMSE, while Danos-8 is the best in terms of MRR.

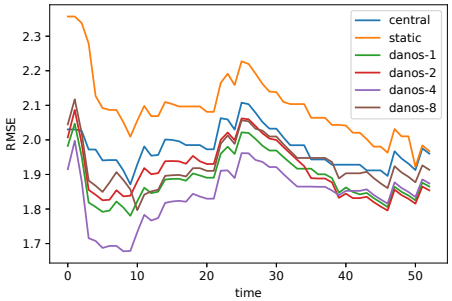

Figure 1: Rolling average of RMSE over time.

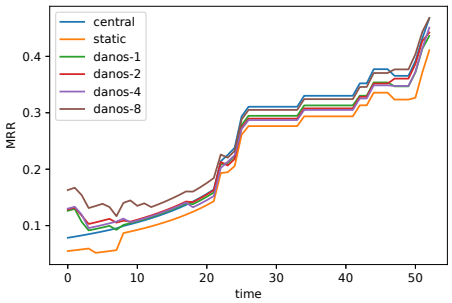

Figure 2: Rolling average of MRR over time.

5 CONCLUSION

The current digital SIoT reality may have proved an utter chaos for the users —- an incomprehensible digital lake of attention, composed of billions of nodes and multi-purpose heterogeneous information sources. In this paper, we argue that the convergence of SIoT and personalized recommender technologies can play a significant role towards the proactive discovery (*what*) and delivery of information and resources *when* they need it most and in a personalized (*how*) manner, increasing the users' experience and the efficiency and effectiveness of their tasks execution. Accordingly, we propose a recommendation engine in SIoT, that runs over an agent-based decentralized architecture, with the aim to create a good pairing between the objects for delivering best fit recommendations to their owners. Main innovation points of the current solution are: the execution is in a fully dynamic social network; there is no full view of the network and historical information; the information distributed among agents is not centrally controlled and respects security and privacy concerns; the network formation is optimized over time leading in better recommendation effectiveness than a centralized approach, as evidenced by experiments in a small-scale dataset. In the future, we plan to evaluate our approach using large-scale datasets.

REFERENCES

[1] Flora Amato, Antonino Mazzeo, Vincenzo Moscato, and Antonio Picariello. 2013. A recommendation system for browsing of multimedia collections in the internet of things. In *Internet of things and inter-cooperative computational technologies for collective intelligence*. Springer, 391–411.

[2] Luigi Atzori, Antonio Iera, and Giacomo Morabito. 2010. The internet of things: A survey. *Computer networks* 54, 15 (2010), 2787–2805.

[3] Luigi Atzori, Antonio Iera, and Giacomo Morabito. 2011. Siot: Giving a social structure to the internet of things. *IEEE communications letters* 15, 11 (2011), 1193–1195.

[4] Luigi Atzori, Antonio Iera, Giacomo Morabito, and Michele Nitti. 2012. The social internet of things (siot)–when social networks meet the internet of things: Concept, architecture and network characterization. *Computer networks* 56, 16 (2012), 3594–3608.

[5] Ranieri Baraglia, Patrizio Dazzi, Matteo Mordacchini, and Laura Ricci. 2013. A peer-to-peer recommender system for self-emerging user communities based on gossip overlays. *J. Comput. System Sci.* 79, 2 (2013), 291–308.

[6] Lucas Nunes Barbosa, Jonathan Gemmell, Miller Horvath, and Tales Heimfarth. 2018. Distributed User-Based Collaborative Filtering on an Opportunistic Network. In *2018 IEEE 32nd International Conference on Advanced Information Networking and Applications (AINA)*. IEEE, 266–273.

[7] Felix Beierle and Tobias Eichinger. 2019. Collaborating with Users in Proximity for Decentralized Mobile Recommender Systems. *arXiv preprint arXiv:1906.03114* (2019).

[8] Zhikui Chen, Ruochuan Ling, Chung-Ming Huang, and Xu Zhu. 2016. A scheme of access service recommendation for the Social Internet of Things. *Int. J. Communication Systems* 29, 4 (2016), 694–706. https://doi.org/10.1002/dac.2930

[9] Cheng Cheng, Chunhong Zhang, Xiaofeng Qiu, and Yang Ji. 2014. The Social Web of Things (SWoT)-Structuring an Integrated Social Network for Human, Things and Services. *JCP* 9, 2 (2014), 345–352.

[10] Daniel Defiebre and Panagiotis Germanakos. 2019. A Human-Centred Business Scenario in SIoT–The Case of DANOS Framework. In *IFIP Conference on Human-Computer Interaction*. Springer, 579–583.

[11] Daniel Defiebre and Panagiotis Germanakos. 2019. Towards a Human-Centered Model in SIoT - Enhancing the Interaction Behaviour of Things with Personality Traits. In *Proc. of the 5th IEEE International Conference on Internet of People*. IEEE.

[12] Alexander Felfernig, Seda Polat Erdeniz, Christoph Uran, Stefan Reiterer, Muesluem Atas, Thi Ngoc Trang Tran, Paolo Azzoni, Csaba Király, and Koustabh Dolui. 2019. An overview of recommender systems in the internet of things. *J. Intell. Inf. Syst.* 52, 2 (2019), 285–309. https://doi.org/10.1007/s10844-018-0530-7

[13] Agostino Forestiero. 2017. Multi-agent recommendation system in Internet of Things. In *2017 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID)*. IEEE, 772–775.

[14] Lewis R. Goldberg. 1990. An alternative" description of personality": the big-five factor structure. *psycnet.apa.org* (1990). http://psycnet.apa.org/journals/psp/59/6/1216.html?uid=1991-09869-001

[15] Samuel Greengard. 2015. *The internet of things*. MIT press.

[16] Peng Han, Bo Xie, Fan Yang, and Ruimin Shen. 2004. A scalable P2P recommender system based on distributed collaborative filtering. *Expert systems with applications* 27, 2 (2004), 203–210.

[17] Anne-Marie Kermarrec, Vincent Leroy, Afshin Moin, and Christopher Thraves. 2010. Application of random walks to decentralized recommender systems. In *International Conference On Principles Of Distributed Systems*. Springer, 48–63.

[18] Jin-Seo Lee and In-Young Ko. 2016. Service recommendation for user groups in internet of things environments using member organization-based group similarity measures. In *2016 IEEE International Conference on Web Services (ICWS)*. IEEE, 276–283.

[19] Zhiting Lin and Liang Dong. 2018. Clarifying Trust in Social Internet of Things. *IEEE Trans. Knowl. Data Eng.* 30, 2 (2018), 234–248. https://doi.org/10.1109/TKDE.2017.2762678

[20] Ibrahim Mashal, Osama Alsaryrah, and Tein-Yaw Chung. 2016. Performance evaluation of recommendation algorithms on Internet of Things services. *Physica A: Statistical Mechanics and its Applications* 451 (2016), 646–656.

[21] Paolo Massa and Paolo Avesani. 2007. Trust-aware recommender systems. 17–24. https://doi.org/10.1145/1297231.1297235

[22] Venus Mohammadi, Amir Masoud Rahmani, Aso Mohammed Darwesh, and Amir Sahafi. 2019. Trust-based recommendation systems in Internet of Things: a systematic literature review. *Human-centric Computing and Information Sciences* 9, 1 (2019), 21.

[23] Francesco Ricci, Lior Rokach, and Bracha Shapira. 2011. Introduction to recommender systems handbook. In *Recommender systems handbook*. Springer, 1–35.

[24] Yasir Saleem, Noel Crespi, Mubashir Husain Rehmani, Rebecca Copeland, Dina Hussein, and Emmanuel Bertin. 2016. Exploitation of social IoT for recommendation services. In *2016 IEEE 3rd World Forum on Internet of Things (WF-IoT)*. IEEE, 359–364.

[25] Zhangyang Wang, Xianming Liu, Shiyu Chang, Jiayu Zhou, Guo-Jun Qi, and Thomas S Huang. 2015. Decentralized recommender systems. *arXiv preprint arXiv:1503.01647* (2015).

[26] Wan-Shiou Yang and San-Yih Hwang. 2013. iTravel: A recommender system in mobile peer-to-peer environment. *Journal of Systems and Software* 86, 1 (2013), 12–20.

[27] Lina Yao, Quan Z Sheng, Anne HH Ngu, and Xue Li. 2016. Things of interest recommendation by leveraging heterogeneous relations in the internet of things. *ACM Transactions on Internet Technology (TOIT)* 16, 2 (2016), 9.

[28] Lina Yao, Xianzhi Wang, Quan Z Sheng, Schahram Dustdar, and Shuai Zhang. 2019. Recommendations on the Internet of Things: Requirements, Challenges, and Directions. *IEEE Internet Computing* 23, 3 (2019), 46–54.

[29] Cai-Nicolas Ziegler. 2005. *Towards decentralized recommender systems*. Ph.D. Dissertation. Albert-Ludwigs-Universität Freiburg.