Adaptation for the Masses: Towards Decentralized Adaptation in Large-Scale P2P Recommenders

Davide Frey, Anne-Marie Kermarrec, Christopher Maddock, Andreas Mauthe, François Taïani

Adaptation for the Masses: Towards Decentralized Adaptation in Large-Scale P2P Recommenders

Davide Frey[1], Anne-Marie Kermarrec[1], Christopher Maddock[2],
Andreas Mauthe[2], François Taïani[3]
[1]Inria, Rennes, France
[2]School of Computing and Communications, Lancaster University, UK
[3]Université de Rennes 1, IRISA – ESIR, France
{anne-marie.kermarrec, davide.frey}@inria.fr,
{c.maddock1,a.mauthe}@lancaster.ac.uk, francois.taiani@irisa.fr

ABSTRACT

Decentralized recommenders have been proposed to deliver privacy-preserving, personalized and highly scalable on-line recommendation services. Current implementations tend, however, to rely on hard-wired, mechanisms that cannot adapt. Deciding beforehand which hard-wired mechanism to use can be difficult, as the optimal choice might depend on conditions that are unknown at design time. In this paper, propose a framework to develop dynamically adaptive decentralized recommendation systems. Our proposal supports a *decentralized* form of adaptation, in which individual nodes can independently select, and update their own recommendation algorithm, while still collectively contributing to the overall system's services.

Categories and Subject Descriptors

I.5.3 [**Clustering**]: Algorithms; D.4.7 [**Organization and Design**]: Distributed systems; K.6.4 [**System Management**]: Centralization/decentralization

General Terms

ALGORITHMS

Keywords

Collaborative filtering, Recommendation systems, Heterogeneity, Adaptivity

1. INTRODUCTION

The Web 2.0 has reshaped the modern-day Internet. Billions of users across the globe access content produced not only by well-known websites, but also by users themselves through collaborative platforms such as online forum or social-networking websites [11]. This results in a spectacular growth in available content, and raises many engineering challenges on how to build architectures that can manage this content in a scalable, resilient, and efficient manner.

Recommendation has emerged as a key service to navigate the resulting data deluge. Recommender systems exploit a user's personal data and past on-line behavior to propose personalized services, resources, and information. Designing scalable and privacy-preserving recommenders is hard, and one promising approach consists in exploiting highly scalable and fully decentralized mechanisms such as gossip protocols [20, 4], or distributed hash tables [28].

These decentralized recommenders have however been limited so far to relatively homogeneous configurations. They typically rely on one similarity metric [29] to self-organize large numbers of users in implicit communities and offer powerful means to search, mine, and compute personalized recommendations. Figuring out the right similarity metric that best fits the needs of a large collection of users is however, highly challenging.

To address this challenge, we explore in this paper how dynamic adaptation can be applied to large-scale decentralized recommenders by allowing each individual nodes to opt autonomously between different topological options. This work seeks to extend early explorations in the field of large-scale peer-to-peer self-adaptation [18], and our early results demonstrate the feasibility of decentralized self-adaptation in peer-to-peer recommender systems.

The paper is organized as follows. Section 2 provides some background and motivates our work. Section 3 presents our proposal. We present our evaluation approach in Section 4. Evaluation results are presented in Section 5, related work in Section 6, and the conclusion in Section 7.

2. BACKGROUND

Recommendation systems have transformed the way we access content on the Internet. Initially introduced to prioritize Usenet news [19], they are becoming an integral part of most online applications: from social networks [11] to online media [27], from e-commerce [21] to news websites [8]. Because of their success, recommendation systems are today hitting the boundaries of traditional system-engineering practices. But, in the vast majority of cases, they remain based around centralized designs.

2.1 Limitations of Centralized Recommenders

Centralization comes with two critical drawbacks. First, centralization raises the specter of a *big brother* society, in

ARM'14 December 9, 2014, Bordeaux, France

DOI: http://dx.doi.org/10.1145/2677017.2677021

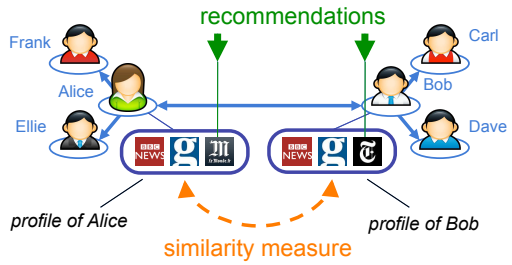

Figure 1: Recommendation in an implicit overlay

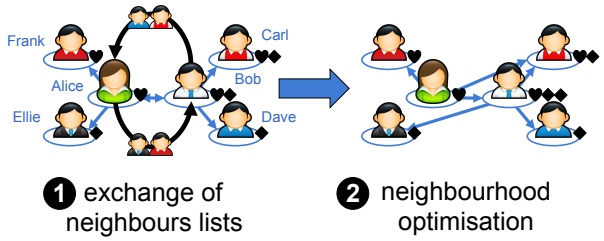

Figure 2: Refining neighborhoods using interests

which a few powerful players are able to gather, process, and analyze large swaths of personal data for their own purposes and with little oversight. This is a threat made very real by recent revelations on NSA activities, or by the news of attacks on personal data stored on cloud servers.

Second, the coexistence of multiple competing centralized systems leads to a 'data siloing' effect. Information about the interests of a user is scattered across a multitude of competing services. This not only makes it impossible for a service to reuse data gathered by another player. But it also makes it very difficult for users themselves to exploit their personal data directly without intermediaries [30].

These crucial limitations have motivated research on decentralized recommendation systems [4, 5, 3, 24].

2.2 Epidemic Decentralized Recommendation

A particularly successful technology for decentralized recommendation consists of implicit interest-based *overlays* [29]. In these overlays each user is associated with an independent computer that stores the user's information, and represents the user in the system. These computers are usually referred to as *nodes* to emphasize their distributed nature. The system then organizes these nodes in a *peer-to-peer topology* (or *overlay*) in which each node is connected to a limited number of other nodes (the node's *neighbors*). (In the following we use *user* and *node* interchangeably.)

These overlays seek to connect users with their most similar other users in the system, according to a *similarity metric* such as profile overlap, Jaccard, or Cosine similarity. The system then uses the resulting k *nearest neighbors* or $k$nn (where k is small) to deliver personalized recommendation in a scalable on-line manner. Figure 1 illustrates this operation. In this example each user is connected through the system to three other users, based on his/her on-line browsing history. *Alice* has been found to be most similar to *Frank*, *Ellie*, and *Bob*; and *Bob* has been found to be most similar to *Carl*, *Dave*, and *Alice*.

Although *Bob* and *Alice* have been detected to be very similar, their browsing histories are not identical: *Bob* has not visited *Le Monde*, but has read the *New York Times*, which Alice has not. The system can use *Bob*'s browsing history to recommend the *New York Times* to *Alice*, and reciprocally use Alice's history to recommend *Le Monde* to Bob, thus providing a form of decentralized collaborative filtering [12].

Gossip algorithms [9, 29] turn out to be particularly useful in building such interest-based overlays. By means of periodic pair-wise information exchanges, they can effectively create clusters of similar users within the system. Users typically start with a random neighborhood, provided by a random peer sampling service [16]. The protocol then runs in *asynchronous rounds*, during which each user repeatedly exchanges information with its neighbors, and seeks to improve its neighborhood in terms of similarity.

For instance, in Figure 2, *Alice* is interested in hearts, and is currently connected to *Frank*, who also likes hearts, and also *Ellie*, who is only interested in diamonds. When *Alice* exchanges her neighbor list with *Bob*, she learns of *Bob's* neighbors, and finds out about *Carl*, who shares her interest in hearts. As such, *Alice* drops *Ellie* from her neighborhood, and replaces her with *Carl*, who will be more useful in providing recommendations to *Alice*.

This greedy sampling procedure is usually complemented by also considering a few random peers (returned by a decentralized *peer sampling service* [16]) as potential neighbors to escape local minima.

2.3 Self-Adaptive Implicit Overlays

The overall performance of a service using a $k$nn overlay critically depends on the quality of the similarity metric it uses. Unfortunately, deciding at design time which similarity metric will work best for all nodes is highly challenging. The same metric might not work equally well for all users [18], and imposing a homogeneous choice to all nodes may be sub-optimal. Further, user behavior might evolve over time, thereby rendering a static metric choice sub-efficient, even though this choice worked well initially.

Despite these limitations, existing decentralized recommendation systems almost invariably use a single similarity metric that is statically selected at design time [4, 2, 3].

The research introduced in this paper investigates the question of whether it is possible to instead both find, and update a node's optimal metric dynamically, *during* the recommendation process. Adapting a node's similarity metric is, however, particularly difficult for at least three reasons. Firstly, in the similarity-based overlays we consider, the nodes only have access to a limited subset of the whole system (i.e. their *neighborhood*) and must make adaptation decisions based on this limited context. This partial knowledge means that it may not be possible to implement a workable adaptation mechanism in such systems, since too much information is missing.

Second, there is a circular dependency between the information available to nodes for adaptation decisions and the actual decision taken. A node must rely on its neighborhood to decide whether a new metric might be preferable to the metric it currently uses. Though this neighborhood depends in turn on the actual metric being used by the node, adding further instability to the adaptation.

Finally, because of the decentralized nature of these systems, nodes should ideally be able to adapt independently from each other, in order to limit synchronization and max-

Algorithm 1 SIMILITUDE's Main Loop by node p

1: **in every round do**
2: $cand \leftarrow \Gamma(p) \cup \Gamma^2(p) \cup 1$ random node
3: ADAPTSIM($cand$)
4: $\Gamma(p) \leftarrow \underset{q \in cand}{\mathbf{argtop}^k} \left(p.\text{sim}\big(\text{items}(p), \text{items}(q)\big)\right)$
5: $it_\Gamma \leftarrow \text{items}\big(\Gamma(p)\big) \setminus \text{items}(p)$
6: $rec \leftarrow \underset{i \in it_\Gamma}{\mathbf{argtop}^m} \left(\text{SCORE}\big(i, p.\text{sim}, \text{items}(p), \Gamma(p)\big)\right)$
7: **end round**
8: **function** SCORE($i, sim, items, \Gamma$)
9: **return** $\sum_{\substack{q \in \Gamma: \\ i \in \text{items}(q)}} sim(items, \text{items}(q))$
10: **end function**

Algorithm 2 Similarity adaptation by node p

1: **function** ADAPTSIM($cand$)
2: $top\_sims \leftarrow \underset{s \in \mathcal{S}}{\mathbf{argmax}} \left(\mathbf{avg}_4\big(\text{EVAL\_SIM}(s, cand)\big)\right)$
3: **if** $p.\text{sim} \notin top\_sims$ **then**
4: $p.\text{sim} \leftarrow$ random element from top_sims
5: **end if**
6: **end function**
7: **function** EVAL_SIM($s, cand$)
8: $hidden_f \leftarrow$ proportion f of items(p)
9: $visible_f \leftarrow \text{items}(p) \setminus hidden_f$
10: $\Gamma_f \leftarrow \underset{q \in cand}{\mathbf{argtop}^k} \left(s\big(visible_f, \text{items}(q)\big)\right)$
11: $it_f \leftarrow \text{items}\big(\Gamma_f\big) \setminus visible_f$
12: $rec_f \leftarrow \underset{i \in it_f}{\mathbf{argtop}^m} \left(\text{SCORE}(i, s, visible_f, \Gamma_f)\right)$
13: **return** $S = \dfrac{|hidden_f \cap rec_f|}{|hidden_f|}$
14: **end function**

imize scalability. The overall convergence of the overlay is, however, an emergent behavior that depends on the collaboration of all nodes. Thus, the central question is: *can nodes adapt independently while preserving the system's overall convergence*?

3. DECENTRALIZED ADAPTATION

3.1 System Model and Overlay Construction

We assume a peer-to-peer system in which each node p possesses a set of *items*, items(p) (in our evaluation: Twitter subscriptions), and maintains a set of k neighbors, where k, the fan-out, is a fixed parameter of the system ($k = 10$ in our experiments). p's neighbors are noted $\Gamma(p)$, and by extension, $\Gamma^2(p)$ are p's neighbors' neighbors. Each node p is associated with a similarity metric, noted p.sim, which takes two sets of items and returns a similarity value.

The main loop of our algorithm (dubbed SIMILITUDE) is shown in Algorithm 1 (when executed by node p). Ignoring line 3 for the moment, lines 2 and 4 implement the greedy $k$nn mechanism presented in Section 2.2. At line 4, $\mathbf{argtop}^k$ selects the k nodes of *cand* (the candidate nodes that may become p's new neighbors) that maximize the similarity expression $p.\text{sim}\big(\text{items}(p), \text{items}(q)\big)$.

Recommendations are generated at lines 5 and 6 from the set it_Γ of subscriptions of all users in p's neighborhood (noted $\text{items}\big(\Gamma(p)\big)$). Recommendations are ranked using the function SCORE at line 8, with the similarity score of the user(s) they are sourced from. Recommendations suggested by multiple users take the sum of all relevant scores. The top m recommendations from it_Γ (line 6) are suggested to the user (or all of them if there are less than m).

3.2 Dynamic Adaptation of Similarity

The adaptation mechanism we propose, ADAPTSIM, is called at the start of each clustering round (line 3 of Algorithm 1) and is shown in Algorithm 2. A node p estimates the potential of each available metric ($s \in \mathcal{S}$ at line 2) using the function EVAL_SIM(s). In EVAL_SIM(s), p hides a fraction f of its own items (lines 8 and 9) and creates a 'temporary potential neighborhood' Γ_f for each similarity metric available (line 10). (We use $f = 20\%$ in our evaluation.) From each temporary neighborhood, the node generates a set of recommendations (lines 11 and 12, see Section 3.1) and evaluates them against the fraction f of internally hidden items. This allows it to associate each similarity s with a score S consisting of the fraction of $hidden_f$ interests matched – 'recall' (line 13).

Node p repeats this evaluation process four times and averages the results (operator $\mathbf{avg}_4$ at line 2 of Algorithm 2). Using these averaged scores, p computes the set of the highest-achieving metrics (*top_sims*). (This is a set as multiple metrics may achieve the same score.) If the current metric-in-use $p.\text{sim}$ is in *top_sims*, p continues to use it; if not, p selects a random metric from *top_sims* (lines 3-4).

After selecting a new metric, a node suspends the metric-selection process for two rounds during which it only refines its neighbors. This *cool-off* period allows the newly selected metric to start building a stable neighborhood thereby limiting oscillation and instability.

4. EVALUATION APPROACH

We validate the adaptation strategies we propose by carrying out simulations on a data set obtained from Twitter. In this section we present our evaluation protocol, before discussing our experimental results in Section 5.

4.1 Data Set

Our evaluation is based on the profiles of 1,000 similarly-geolocated Twitter users, randomly selected from the larger data set presented in [7][1]. The profile of each user is composed of their twitter subscriptions (*feeds*). All feeds known to less than 20 users were removed, as these feeds are of little interest to the collaborative filtering process. Finally, all users with less than five remaining subscriptions were also removed. Once this process was complete, 833 users remained, with a mean of 66 subscriptions per user.

4.2 Evaluation Metrics

To evaluate the success of the experiment, we record four metrics: Recall, Precision, Number of 'Optimal Metrics', and Level of Instability.

A recommendation system offers two main points of evaluative interest: the number of valid recommendations which can be made, and the accuracy of the recommendation set as a whole. These two aspects are measured using *recall*

[1]An anonymized version of this data set is available at `http://ftaiani.ouvaton.org/ressources/onlyBayLocsAnonymised_21_Oct_2011.tgz`

(measuring the ability of the system to return many correct recommendations) and *precision* (measuring the ability of the system to return few incorrect recommendations).

Alongside these two general metrics for recommendation systems, we also use two more specific metrics. We first count how many nodes reach their *optimal* similarity metrics. We define more precisely what we understand by *optimal* in Section 4.5. Finally, we observe the level of instability within the system, by recording the number of nodes switching metric during each round.

4.3 Simulator and Cross Validation

We use cross-validation to measure the quality of the recommendations returned by our system. We simulate our system using 80% of user's profiles (visible data set) to construct the similarity-based overlay and compute recommendations, and use the remaining 20% (hidden data set) to calculate the quality of these recommendations.

Our simulations use the following parameters: Each node is randomly assigned a neighborhood of 10 nodes, as well as a randomly selected similarity metric to start the refinement process. Simulations run for 200 rounds. 66 suggestions are returned to each node in each round, and the recall and precision metrics are computed based on these suggestions. We use two rounds of cool-off by default.

We repeat each simulation 10 times and average the results. After completing each set of 200 rounds, all nodes clear their data structures and the simulation restarts.

4.4 Similarity Metrics

Our evaluation uses four similarity metrics: *Overlap*, *Big*, *OverBig* and *Jaccard* [18], shown in Fig. 3. These metrics are sufficiently different to represent distinct similarity choices for each node, and offer a representative adaptation scenario.

Overlap is a simple count of the subscriptions shared by a user and its neighbor. As such, it would have the tendency to favor users with a large number of subscriptions. *Big* simply counts the number of subscriptions of the neighbor, presuming that the greater the number of subscriptions available, the more likely a match is to be found somewhere in the list. This likewise favors users with a larger number of subscriptions. *Overbig* works by combining Big and Overlap — which allows the least similar high-subscription users to be discredited. Finally *Jaccard* makes the overlap of subscriptions relative to the total number of subscriptions of the two users, and thus, provides improved results for users with fewer subscriptions.

It is important to note that the actual set of metrics is not our main focus. Rather, we are interested in the adaptation process, and seek to demonstrate how recommendations can be improved by adjusting nodes' similarity metrics.

4.5 Static Metrics Allocations

We compare our approach to *five* static (i.e. non-adaptive) system configurations, which serve as baselines for our evaluation. In the first four baseline configurations, we statically allocate the same metrics to all nodes from the set of possible metrics defined in Fig. 3 (*Overlap*, *Big*, *OverBig*, and *Jaccard*). These baselines are static and homogeneous.

$$
\begin{aligned}
Overlap(u_i, u_j) &= |subs_i \cap subs_j| \\
Big(u_i, u_j) &= |subs_j| \\
OverBig(u_i, u_j) &= Overlap(u_i, u_j) + Big(u_i, u_j) \\
Jaccard(u_i, u_j) &= \frac{Overlap(u_i, u_j)}{subs_i \cup subs_j}
\end{aligned}
$$

Figure 3: The four similarity metrics used

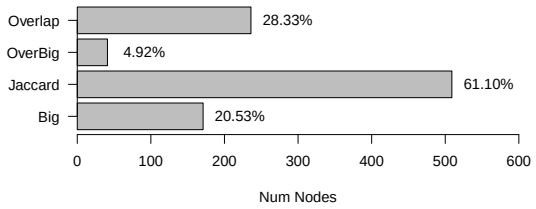

Figure 4: Distribution of optimal metrics

The fifth (HeterRand) randomly allocates one of the four above metrics to each node. This configuration corresponds to a situation in which the system has no a-priori knowledge regarding optimal metrics, and does not use dynamic adaptation.

5. EXPERIMENTAL RESULTS

5.1 Static Baseline

The distribution of optimal metrics obtained by individual nodes (Sec. 4.5) is shown in Fig. 4. To estimate variability, the entire experiment is repeated twice, and the two sets of results compared node by node. In total, 88.2% of nodes report the same list of optimal metrics across both tests. Of the 98 nodes that did not, 20% listed optimal metrics which were a subset of those they had found in the first run, and all but one of this 20% had under 30 subscriptions. (Well below the mean of 66.) 42 nodes were unable to settle on a single optimal metric, all but one of those recording no difference between any of the four metrics. These nodes were generally those with the smallest amount of data available.

The distribution in Fig. 4 diverges from that of [18] as we did not use the exact same dataset. This highlights the difficult of selecting a static metric at a system's design time.

5.2 Similitude

We test Similitude with a cool-off period of two rounds. We first evaluate the convergence of recall and precision for Similitude and compare it with that of HeterRand (Figures 5 and 6). We then analyze the number of users that select one of the optimal metrics as discussed in Sec. 5.1 (Fig. 7), and the switching activity of users (Fig. 8). Table 1 completes these results with the recall and precision obtained at round 200 (converged state), while Table 2 presents converged-state results for non-adaptive solutions exploiting each predefined static metric.

Results show that Similitude allows nodes to both find their optimal metric and switch to it. Compared to a static random allocation of metrics (HeterRand), Similitude improves recall by 17.8% (from 0.28 to 0.34), and precision by 22% (from 0.038 to 0.046) as shown in Table 1.

Table 2 shows that the average performance of non-adaptive metrics is very close to that of HeterRand. Similitude outperforms all but one of these static homogeneous metrics (*Jaccard*, which shows a recall of 0.39 compared to 0.34). However, selecting *Jaccard* statically would require knowing that this metric performs best, a knowledge that is often not available in evolving systems. As users' behaviors

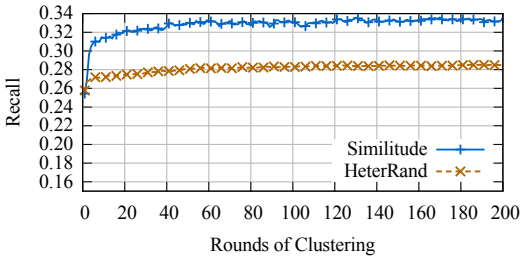

Figure 5: Recall

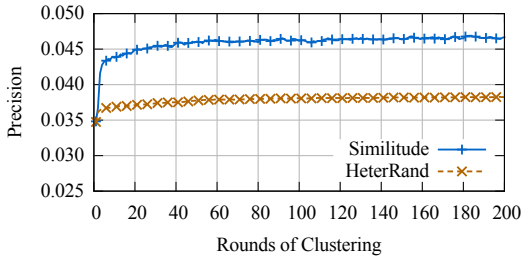

Figure 6: Precision

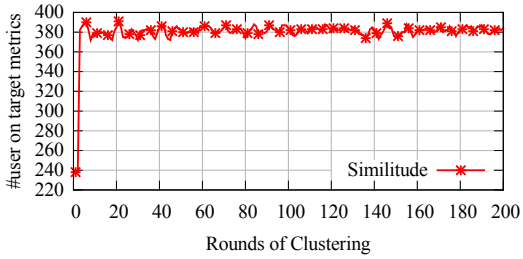

Figure 7: Users selecting their optimal metrics

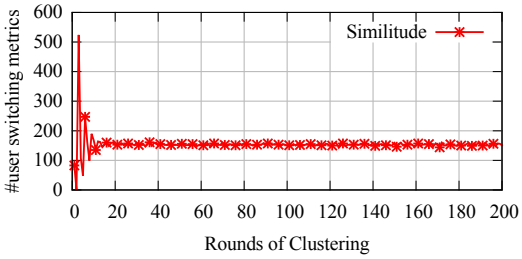

Figure 8: Switching activity with Similitude

evolve, the high gain of *Jaccard* might be replaced by that of another metric. Similitude makes it instead possible to achieve stable performance in spite of dynamic behavior.

5.3 Discussion

Despite these encouraging results, SIMILITUDE opens several directions for improvement. First, Fig. 7 shows that only 46.1% of nodes (384 out of 833) eventually switch to an optimal metric. This leads to the question of why so many nodes are settling for a sub-par metric.

similarity	recall	gain	precision	gain
HeterRand	0.2850	—	0.0383	—
Similitude	0.3358	17.82%	0.0467	22.04%

Table 1: Gain of Similitude over HeterRand

similarity	recall	gain	precision	gain
Jaccard	0.3867	35.70%	0.0517	35.16%
Overlap	0.3452	21.11%	0.0449	17.33%
Big	0.2075	-27.19%	0.0283	-25.91%
OverBig	0.1984	-30.40%	0.0278	-27.35%
Average	**0.2844**	**-0.19%**	**0.0382**	**-0.19%**

Table 2: Evaluating static homogeneous metrics. The gain columns show the gains of each row over the baseline (HeterRand)

A second improvement direction results from Fig. 8: while most nodes choose their favorite metric within 20 cycles, approximately 20% of them never settle for any metric. To address this instability, we are investigating solutions that average the scores obtained by a metric over multiple rounds.

Finally, we observe that a node chooses its metric based on its current neighborhood. This tends to favor the metric that created this neighborhood, even if another one might yield better results on a different neighborhood. We are thus evaluating a model in which nodes evaluate metrics against randomly selected neighborhoods.

6. RELATED WORK

Several research efforts have recently concentrated on decentralized recommenders [13, 23, 1, 6, 26] to investigate their advantages in terms of scalability and privacy. Earlier approaches exploit distributed hash tables (DHTs) in the context of recommendation. For example, PipeCF [13] and PocketLens [23] propose a Chord-based CF systems to decentralize the recommendation process on a P2P infrastructure. Yet, more recent solutions have concentrated on the use of randomized and gossip-based protocols [4, 17, 3].

Recognized as a fundamental tool for information dissemination [15, 22], Gossip protocols exhibit innate scalability and resilience to failures. As information is generally copied over many links, a single lost connection generally has no effect on information dissemination. Yet, their probabilistic nature also makes gossip protocols particularly suited to applications involving uncertain data, like recommendation.

Olsson's *Yenta* [25] was one of the first systems to employ gossip protocols in the context of recommendation. This theoretical work enhances decentralized recommendation by taking trust between users into account. The *Gossple* system [4] uses a similar theory to enhance navigation through query expansion and was later extended to news recommendation [5]. Finally, in [14], Hegedűs et al. present a gossip-based learning algorithm that carries out 'random walks' through a network to monitor concept drift and adapt to change in P2P data-mining.

The adaptive element of our work is partially motivated by [18], where the authors demonstrate that higher-quality recommendation is achievable when multiple different similarity metrics are used in a heterogeneous fashion, across a network. Contrary to the approach we have presented, [18] exploits a static a-priori selection of metrics, and does not propose any mechanism to select metrics dynamically. Other related work has looked at matching events against interests in pub-sub infrastructures, and considered how the infrastructure could use self-adaptation within an overlay network, to adapt to changes in the application traffic [10].

7. CONCLUSION

We demonstrated the viability of an adaptive, decentralized recommendation system that exploits a variety of similarity metrics. Our results indicate that such a system can offer an improvement over existing homogeneous methods.

The adaptation algorithm we have presented remains basic: we plan to investigate how the choice of adaptation could be further refined, for instance by taking into account the decisions of neighboring nodes, or by trying to minimize the advantage of the current metrics over its competitors. Also it would be interesting to evaluate a practical implementation of such a network. Our study exhibits limitations resulting from the characteristics of the data set we used (sparsity and small size). Sparsity is also likely to occur in a real implementation. However, running the system with a larger, less refined data set may provide an interesting insight into more potential problems the system may face.

Acknowledgments

This work has been partially funded by the French National Research Agency (ANR) project *SocioPlug* under contract ANR-13-INFR-0003 (`http://socioplug.univ-nantes.fr`).

8. REFERENCES

[1] Tribler. `http://www.tribler.org`.

[2] X. Bai, M. Bertier, R. Guerraoui, A.-M. Kermarrec, and V. Leroy. Gossiping personalized queries. In *EDBT'2010*.

[3] R. Baraglia, P. Dazzi, M. Mordacchini, and L. Ricci. A peer-to-peer recommender system for self-emerging user communities based on gossip overlays. *J. of Comp. and Sys. Sciences*, 2013.

[4] M. Bertier, D. Frey, R. Guerraoui, A.-M. Kermarrec, and V. Leroy. The gossple anonymous social network. In *Middleware'2010*.

[5] A. Boutet, D. Frey, R. Guerraoui, A. Jégou, and A.-M. Kermarrec. WhatsUp Decentralized Instant News Recommender. In *IPDPS*, 2013.

[6] A. Boutet, D. Frey, G. Rachid, A. Jégou, and A.-M. Kermarrec. Privacy-Preserving Distributed Collaborative Filtering. In *NETYS*, Marrakech, Morocco, May 2014.

[7] J. Carretero, F. Isaila, A.-M. Kermarrec, F. Taïani, and J. M. Tirado. Geology: Modular georecommendation in gossip-based social networks. In *ICDCS 2012*.

[8] A. S. Das, M. Datar, A. Garg, and S. Rajaram. Google news personalization: scalable online collaborative filtering. In *WWW*, 2007.

[9] A. Demers, D. Greene, C. Hauser, W. Irish, J. Larson, S. Shenker, H. Sturgis, D. Swinehart, and D. Terry. Epidemic Algorithms for Replicated Database Maintenance. In *Proc. of PODC*, 1987.

[10] E. Di Nitto, D. J. Dubois, and A. Margara. Reconfiguration primitives for self-adapting overlays in distributed publish-subscribe systems. In *SASO 2012*.

[11] Facebook Inc. Facebook: Company info – statistics. `https://newsroom.fb.com/company-info/`, March 2014. Accessed: 2014-05-13.

[12] D. Goldberg, D. Nichols, B. M. Oki, and D. Terry. Using collaborative filtering to weave an information tapestry. *CACM*, 1992.

[13] P. Han, B. Xie, F. Yang, and R. Shen. A scalable p2p recommender system based on distributed collaborative filtering. *Expert Systems with Applications*, 2004.

[14] I. Hegedus, R. Ormándi, and M. Jelasity. Gossip-based learning under drifting concepts in fully distributed networks. In *SASO 2012*.

[15] M. Jelasity, A. Montresor, and O. Babaoglu. T-man: Gossip-based fast overlay topology construction. *Computer networks*, 53(13):2321–2339, 2009.

[16] M. Jelasity, S. Voulgaris, R. Guerraoui, A.-M. Kermarrec, and M. van Steen. Gossip-based peer sampling. *ACM TOCS*, 25, 2007.

[17] A.-M. Kermarrec, V. Leroy, A. Moin, and C. Thraves. Application of random walks to decentralized recommender systems. In *OPODIS'10*, Berlin, Heidelberg, 2010. Springer-Verlag.

[18] A.-M. Kermarrec and F. Taïani. Diverging towards the common good: heterogeneous self-organisation in decentralised recommenders. In *SNS'2012*.

[19] J. A. Konstan, B. N. Miller, D. Maltz, J. L. Herlocker, L. R. Gordon, and J. Riedl. Grouplens: Applying collaborative filtering to usenet news. *Commun. ACM*, 40(3):77–87, Mar. 1997.

[20] V. Leroy, B. B. Cambazoglu, and F. Bonchi. Cold start link prediction. In *KDD'2010*.

[21] G. Linden, B. Smith, and J. York. Amazon. com recommendations: Item-to-item collaborative filtering. *IEEE Internet Computing*, 2003.

[22] G. Mega, A. Montresor, and G. P. Picco. Efficient dissemination in decentralized social networks. In *IEEE P2P 2011*.

[23] B. N. Miller, J. A. Konstan, and J. Riedl. Pocketlens: toward a personal recommender system. *TOIS*, 2004.

[24] A. Moreno, H. Castro, and M. Riveill. Decentralized recommender systems for mobile advertisement. In *Workshop on Personalization in Mobile Applications (PEMA'11)*, , Chicago, Illinois, USA, Oct. 2011. ACM.

[25] T. Olsson. Decentralised social filtering based on trust. In *AAAI-98 Recommender Systems Workshop*.

[26] V. Schiavoni, E. Rivière, and P. Felber. Whisper: Middleware for confidential communication in large-scale networks. In *ICDCS 2011*, June 2011.

[27] Y. Song, S. Dixon, and M. Pearce. A survey of music recommendation systems and future perspectives. In *The 9th International Symposium on Computer Music Modeling and Retrieval (CMMR)*, 2012.

[28] J. M. Tirado, D. Higuero, F. Isaila, J. Carretero, and A. Iamnitchi. Affinity p2p: A self-organizing content-based locality-aware collaborative peer-to-peer network. *Comp. Net.*, 54, 2010.

[29] S. Voulgaris and M. v. Steen. Epidemic-style management of semantic overlays for content-based searching. In *Euro-Par'05*.

[30] C.-m. A. Yeung, I. Liccardi, K. Lu, O. Seneviratne, and T. Berners-Lee. Decentralization: The future of online social networking. In *W3C Workshop on the Future of Social Networking*, 2009.