Towards Comprehensive Social Sharing of Recommendations: Augmenting Push with Pull

Harsha V. Madhyastha Megha Maiya
UC Riverside

ABSTRACT

On today's online social networks, a user can discover only those recommendations that her friends put in the effort to share. Therefore, we present the *PullRec* framework for enabling users to *pull* recommendations from their friends. *PullRec* employs two measures to minimize the effort involved in sharing recommendations. First, to reduce the onus on users to express their recommendations, *PullRec* proactively logs all the entities about which a user may have an opinion and attempts to infer the user's opinions. Second, to ensure that users are not spammed with irrelevant queries, when a user queries for recommendations on a certain topic, *PullRec* notifies only those friends of the user who are likely to have relevant recommendations. *PullRec* is a step towards enabling a user to discover all recommendations that her friends are willing to share with her.

Categories and Subject Descriptors

C.2.4 [**Computer Systems Organization**]: General—*System architectures*

Keywords

Social networks, Recommendations

1 Introduction

Today, many online social networks (OSNs) offer primitives that simplify the task of sharing recommendations with one's friends, e.g., Facebook and Google+ offer the 'Like' and '+1' buttons. Application providers (e.g., websites, smartphone apps) can bundle these primitives with their content, so as to enable any user to share her recommendations with her friends. As evidence of the utility of such social sharing of recommendations, Netflix spent $1.3 million lobbying the US government [2] to make it legal for the videos seen by a user to be shared with the user's friends.

Hotnets '13, November 21–22, 2013, College Park, MD, USA.

However, empirical evidence suggests that most users do not choose to use even these seemingly easy-to-use primitives to share their recommendations. For example, on the Google Play store, even for Android apps with overwhelmingly positive reviews, the median app has less than 5% of its users recommending it to their friends. Thus, in contrast to the conventional viewpoint that users are "oversharing" [3] on OSNs, our thesis is that there is significant "undersharing" of recommendations.

In fact, we argue that the extent to which today's OSNs can facilitate sharing of recommendations is fundamentally limited due to the cognitive overhead associated with sharing recommendations. There are three sources of this overhead. First, when a user likes something, she may be comfortable sharing her recommendation with only a subset of her friends; this subset may vary based on what the user is recommending. Therefore, the act of sharing a recommendation requires the user to carefully think through her privacy preferences. Second, all content is not relevant to all users; if a user were to share everything that she likes with all of her friends, the user risks being perceived as "spamming" her friends. Therefore, the onus again falls on the user to determine which subset of her friends may find her recommendation relevant. In fact, uncertainty about who may find a particular recommendation relevant further dis-incentivizes the user from incurring the mental overhead of making privacy-preserving judgements. Lastly, users often discover entities that they visit in the real-world (e.g., restaurants, vacation spots, movies) via online sources (e.g., Yelp, TripAdvisor, IMDB). To recommend such an entity, users need to remember to go back to the relevant online source at which they had discovered the entity. Users can often forget to do so.

To address these overheads associated with the status quo for sharing recommendations, we make the case for augmenting existing primitives for *pushing* out one's recommendations to friends with *PullRec*—a framework that adds support for *pulling* recommendations from friends. Enabling users to pull recommendations has several advantages over a push-based framework. First, every user can pull recommendations relevant to her, thus offloading the decision process of determining relevance from those producing recommendations. Second, a user needs to think through her privacy

preferences for sharing a particular recommendation only when a friend who finds it relevant asks for it. Finally, even if a user likes a particular entity, the user needs to explicitly express her recommendation of that entity only when a friend asks for her opinion about it.

However, development of a pull-based recommendation sharing framework is associated with several inherent challenges. First, to maximize the chances of discovering all relevant recommendations, a user should attempt to pull information from all of her friends. However, notifying all friends whenever a user is seeking recommendations on a certain topic will likely result in query overload; for most queries, only a small fraction of a user's friends are likely to have recommendations relevant to the query. Second, in contrast to users pushing out recommendations to their friends, a pull-based sharing framework introduces delays (when a user issues a query, she needs to wait for her friends to respond) and impacts privacy (a user's query is revealed to her friends) for those seeking recommendations. Finally, when users are required to issue queries to seek recommendations, they can only receive recommendations that they explicitly seek, thus eliminating the serendipitous discovery possible on push-based recommendation sharing platforms.

In this paper, we present a first attempt at designing *PullRec* in a manner that tackles the above challenges. Our design offers several desirable properties: 1) when a user issues a query, *PullRec* notifies only that subset of the user's friends who potentially have an opinion relevant to the query, 2) *PullRec* minimizes the overhead that users incur in responding to friends' queries, and 3) it balances the trade-off between preserving the privacy of users who issue queries and enabling serendipitous discovery of recommendations. To offer these properties, we leverage several recent trends: our increasing reliance on digital sources of information, the widespread adoption of GPS-enabled smartphones, the commoditization of brainwave sensors, and the use of friend aggregation constructs (such as circles on Google+ and friend lists on Facebook) for sharing information on OSNs.

2 Motivation

Our call for enabling users to pull recommendations from their friends is motivated by the fact that users often do not share their recommendations on today's OSNs. Here, we highlight the potential causes for this undersharing using data from two OSNs: Twitter and Google+.

Sharing has overheads. Twitter offers two primitives for users to express their endorsement of tweets—Favorite and Retweet. When a user Retweets a tweet, she shares this tweet with all of her followers, whereas when a user Favorites a tweet, only the poster of the tweet is notified of this activity, but not the user's followers.

We analyze the relative use of the Favorite and Retweet primitives on the tweets posted by 100 popular news accounts [1]. We focus on these accounts because most (> 85%) of the tweets that they post include links to web pages.

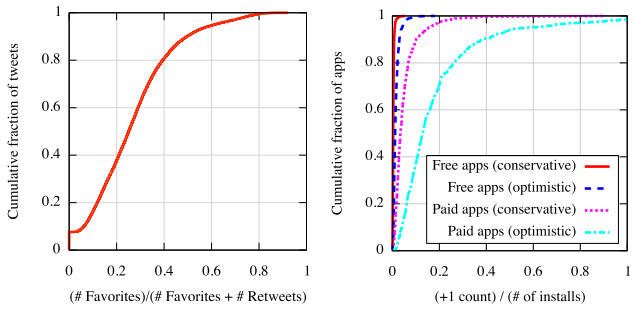

Figure 1: Lack of sharing on (a) Twitter and (b) Google+.

Therefore, users are more likely to like a tweet because of the content included at the web page linked from the tweet, rather than the content of the tweet itself. In late June 2013, we crawled all the 260K tweets containing URLs posted during May 2013 by the 100 monitored accounts. We do not consider recently posted tweets so as to discard cases where a user Favorites a tweet as a way of bookmarking the linked web page for later reading and un-Favorites the tweet once she has done so—a common practice on Twitter.

From this dataset of 260K tweets, we consider a subset of "popular" tweets that many users endorse—ones for which the number of Favorites and Retweets sum to greater than 100. For these popular tweets, Figure 1(a) plots the ratio of the number of Favorites to the number of users who endorsed the tweet in the form of either a Favorite or a Retweet. We see that, for 20% of these popular tweets, over 40% of the users who liked the web page associated with the tweet chose not to recommend this to their followers.

While we can only speculate on the reason many users choose to Favorite these tweets instead of Retweeting them, we believe that the cause for this phenomenon is the relative overhead associated with Retweeting as compared to marking a tweet as a Favorite. To Favorite a tweet, a user only needs to decide whether it is acceptable to notify the poster of the tweet. In contrast, when Retweeting, a user needs to weigh the costs of doing so both with regards to protecting his privacy and determining its relevance for his followers.

Users are lazy/forgetful. On the Google Play store for Android apps, the web page for every app includes a +1 button which users can click to recommend the app to their friends on Google+. Users who +1 an app can share this information with any subset of their circles on Google+.

To understand how common it is for users to use the +1 button to share recommendations of apps, we examine the top 60 apps in every app category on Google Play. For each app, we use three sources of information available on the web page for that app: 1) the number of users who have installed the app (Google only presents a range that encompasses the number of installs), 2) the number of users who clicked the +1 button on the app's web page, and 3) the average rating in reviews posted by users. We consider apps with overwhelmingly positive reviews, i.e., average rating of at least 4.5 on a scale of 5. For every app in this subset, we compute both conservative and optimistic estimates—based

on the upper and lower bounds for the number of installs—for the fraction of the app's users who recommended it to their friends. Figure 1 shows that, even based on optimistic estimates, for over 90% of popular free apps, less than 5% of those who installed the app chose to +1 it. Though the fraction of users who click on the +1 button is higher for paid apps, the optimistic estimate shows that less than 20% of users do so for over 70% of apps.

Given that the reviews for all the considered apps are overwhelmingly positive, and since many other apps do have a sizeable fraction of negative reviews, we believe it to be unlikely that all the users who installed one of these popular apps, but did not click +1, did not like the app. Instead, we speculate that the more likely explanation for the disparity between the number of installs and the number of +1s for most of these apps is that a vast majority of users failed to come back to click +1 on the web page for the app after installing and using the app. This is either because these users simply forgot to return to the web page and click +1, or they were too lazy to do so since it is unclear if any of their friends will find their recommendation relevant.

3 Overview of *PullRec*

Shortcomings of existing options. Say a user is seeking a recommendation on a certain topic, e.g., *"a Thai restaurant in San Francisco"* or an *"an article that explains Bitcoin"*. The user's first option is to search through the recommendations previously shared by her friends. However, our results from Section 2 show that, even if a friend has a relevant recommendation, he may not have shared it due to the effort involved in doing so. Therefore, the next option is for the user to selectively reach out (e.g., via email) to those who the user believes may have recommendations relevant to her needs. In many cases, the user may however not know which subset of friends to contact. For example, the user may not know which of her friends frequent Thai restaurants or which of them have read articles about Bitcoin. Hence, by contacting only a few friends, the user may miss out on relevant recommendations. To maximize the chances of discovering all relevant recommendations, the user must therefore query all of her friends to whom she is comfortable revealing her query, e.g., by posting the query on her OSN profile and sharing this with the appropriate subset of her friends. However, by doing so, not only does the user have to reveal her query even to those friends who have no relevant information to offer, but some of those who have relevant information may not respond [13] (due to the overhead of recalling and responding to the user's query).

***PullRec* architecture.** Thus, all the options currently available to a user for discovering recommendations fall short of enabling comprehensive sharing, i.e., even if one of the user's friends has a recommendation relevant to the user's needs and the friend is comfortable sharing this recommendation with her, the user may not be able to discover that recommendation. To fix this shortcoming in existing options,

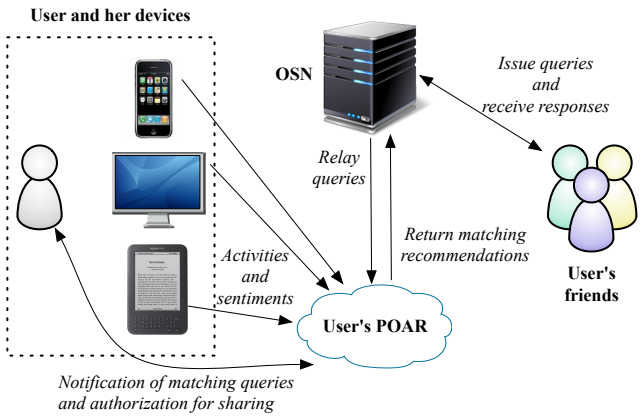

Figure 2: Overview of *PullRec* architecture.

our proposed *PullRec* framework offers a new primitive for discovering recommendations: it enables a user to pull recommendations from all of the user's friends, but notifies only those who have information relevant to the user's needs.

PullRec achieves this goal using the architecture shown in Figure 2. First, every user installs client software that continually monitors the user's activities on all of her devices (similar to the quantified self effort [14] for monitoring a user's health). This monitoring includes keeping track of the information that the user consumes, the locations that the user visits, and the user's sentiments during these activities. The monitoring software installed on a user's devices transmits the information that it gathers to a Personal Opinion and Activity Repository (*POAR*) that every user maintains in the cloud. To protect users' privacy, only the owner of a *POAR* is privy to all the information that it stores, and none of this information is shared with others without the explicit consent of the owner. This is unlike previous proposals (e.g., [12]) which attempt to automatically recommend entities consumed by a user to the user's friends, and have to worry about protecting the user's privacy in doing so.

To enable users to discover recommendations, the OSN presents an interface via which users can query for and view their friends' opinions about items of their interest. When a user queries for her friends' recommendations about a particular topic, the OSN communicates this query to the *POAR* of each of the user's friends. If a friend's *POAR* finds that it has information that is of relevance to the user's query, then that *POAR* notifies its owner in order to seek her permission for sharing this relevant information with the user who issued the query. The *POAR* then shares the relevant information only if its owner consents to the sharing.

Benefits. *PullRec* uses this architecture to offer several desirable properties.

- **Automated recommendation inference:** Given the inherent laziness and inconsistent memory of users, *PullRec* attempts to automate the inference of a users' recommendations where possible. This reduces the need for users to remember what they would like to recommend and share with their friends.

- **Decoupling expression of opinion and sharing:** In current systems, a user's endorsement of a particular entity is tightly coupled with her sharing of this recommendation; when the user expresses her endorsement of an entity, she has to choose whom to share this information with *at that moment*. In contrast, *PullRec* decouples these two functions. *PullRec* lets the user explicitly express her opinion about a particular entity (when it is unable to infer this), and express her privacy preferences for sharing this information at different points in time (only when required).
- **Offloading of relevance decisions and fewer privacy decisions:** *PullRec* frees users from having to determine relevance when sharing recommendations with their friends, since a friend will only query for recommendations that are relevant to him. Moreover, users are required to make privacy decisions on a particular recommendation only when someone who finds it relevant is requesting it.
- **Better privacy/lower spam:** When a user has a query, *PullRec* notifies only that subset of the user's friends who may have a relevant recommendation. In contrast to simply notifying a user's query to all of the user's friends, *PullRec* thus offers better privacy for the user (her query is not revealed to friends who have no relevant information) and lower spam for her friends (a user is not inundated with irrelevant queries for which he has no input to offer).

In the next two sections, we describe the challenges associated with the development of the *PullRec* framework that satisfies these properties and present potential solutions.

4 Proactive logging of user activity

To enable users to *pull* information from their friends' *POARs*, *PullRec* continually logs two types of information to any user's *POAR*: 1) what entities does the user consume?, and 2) what are the user's opinions about these entities? We next discuss how *PullRec* collects both types of information.

4.1 Logging entities consumed

PullRec can potentially infer all entities that a user consumes by recording everything that the user sees and hears, and by applying video and audio processing to this data. However, most users are unlikely to be able to afford the amount of storage and computational resources necessary to store and process data at such scale.

Therefore, to make *PullRec*'s inference of the entities consumed by a user more efficient, we focus on two broad categories of entities—digital entities and real-world entities.

Digital entities include various types of information—web pages, applications, songs, videos, e-books, etc.—that a user consumes via electronic devices that she owns, such as laptops, desktops, smartphones, tablets, audio/video players, and e-book readers. Here, we consider all electronic devices that enable both 1) logging of the user's activity, and 2) relaying of the logged activity to the user's *POAR*. Information logged on a device may be relayed either directly over the network to the user's *POAR* (e.g., smartphones and tablets) or indirectly via another device (e.g., information from iPods and Kindles can be transmitted after syncing with the user's laptop/desktop).

The key challenge in logging the digital entities consumed by a user is that, in order to serve queries for recommendations from friends, it is insufficient to merely log the entities that the user consumes. *PullRec* also needs to annotate every entity with appropriate metadata that enables matching the entity with user queries. For example, if a user reads an article on the Web or listens to a song on her iPod, logging only the article's URL or the song's title will be insufficient to serve queries such as *"Do you recommend any article that explains Bitcoin?"* or *"What recent song would you recommend?"*. To match such queries to the entities consumed by a user, the user's *POAR* needs to store the topic of every web page and the fact that what the user listened to was a song.

PullRec tackles this challenge in two ways. First, every source of information must be tagged with the type of information consumed from it, e.g., a device such as Kindle is tagged with "books" and an app such as Pandora is tagged with "songs". Once a particular information source is tagged, this metadata can be reused across all users. Second, *PullRec* leverages the descriptive metadata that content producers bundle with their content. For example, many web pages embed metadata into their HTML for the benefit of web crawlers (so that they know how to categorize the web page) and OSNs (so that they know what information, in addition to the web page's URL, to share with the user's friends when she clicks Like [4] or +1 [6] on the page). This metadata typically conforms to a well-defined schema (e.g., the schemas published at `http://schema.org`) that has been standardized by major search engines.

Real-world entities include places that the user visits in the real-world. Unlike digital entities, the primary challenge in this case is in even identifying the entities that the user consumes. To serve a friend's request for restaurant recommendations, how can *PullRec* identify the restaurants that a user has visited?

To identify the real-world entities visited by a user, *PullRec* leverages the increasing ubiquity of GPS-enabled smartphones. The *PullRec* client installed on a user's smartphone can use the GPS on the device to continually track the user's location. Smartphone apps such as Google Now are evidence that this location tracking can be performed in an energy-efficient manner. Given a stream of (latitude, longitude) coordinates for a user, *PullRec* can map these locations to real-world entities by using, for example, Google's reverse geocoding API [8].

4.2 Inferring user opinions

As a user's activities are logged to her *POAR*, all the entities consumed by the user cannot simply be shared with all of the user's friends (e.g., as the recommendation system on web-services such as Amazon do). A user may not be in approval of many of these entities, and hence, a user's consumption

of an entity cannot be assumed to indicate the user's recommendation. Moreover, a user may wish to keep many of her consumption activities private, and not share them with some or all of her friends.

Therefore, *PullRec* keeps all the information stored in a user's *POAR* as private to the user and requires her friends to issues queries to access information of relevance to them. When one of the user's friends issues a query, *PullRec* can determine if the user may have relevant recommendations based on the information logged in the user's *POAR* about the entities that she has consumed.

However, once *PullRec* identifies the entities matching a friend's query, it cannot simply ask the user which of those entities she would like to recommend. Since the user had consumed the matching entities at an earlier point in time, some effort may be required on her part to recall her opinion when she is asked for it. The greater the effort required from a user to respond to a query, the lesser the chances of the user responding.

Hence, *PullRec* strives to log a user's opinion about entities *as she is consuming them*, rather than asking her opinion when her response is required to serve a friend's query. *PullRec* combines three techniques for doing so.

Inferring likes from repeated activity. Since *PullRec* continually monitors a user's activities, it can infer that the user likes a particular entity if she consumes it repeatedly. For example, in the case of digital entities, *PullRec* can infer that the user likes a particular song if she repeatedly listens to it. In the case of real-world entities, *PullRec* can similarly infer that the user likes a particular restaurant if she frequents the place several times.

Inferring opinions from user sentiment. However, a user may also wish to recommend many entities that she consumes only once, e.g., a book that she read or a restaurant that she visited while on vacation. In these cases, we envision that *PullRec* can infer the user's opinion about a particular entity she consumes by monitoring the user's brainwaves while she is consuming the entity [11, 9]. Commodity brainwave sensors cost as little as $100 now, e.g., Neurosky's MindWave [7]. If wearable technologies such as Google Glass gain widespread adoption in the future, brainwave sensors can be bundled with these devices without significantly inflating the cost of these devices.

Decoupling recommendation from sharing. The above two techniques may however not suffice to infer the user's opinion in all cases; *PullRec* may not be able to infer the user's opinion about an entity that the user consumes only once either due to the absence of brainwave sensors or because analysis of the user's brainwaves may be inconclusive. Therefore, *PullRec* also allows for the user to explicitly express her opinion about any entity that she consumes. However, unlike existing primitives such as the Like and +1 buttons, *PullRec* decouples the user's expression of her recommendation of an entity from her selection of with whom she would like to share this recommendation. *PullRec* presents the user with an interface whereby the user can express her endorsement of the entity that she is currently consuming. When the user uses this interface to endorse a particular entity, this information is merely logged to the user's *POAR*; sharing decisions are deferred for later.

5 Dissemination of recommendations

All the information about a user's activity and inferred opinions that is logged by *PullRec* cannot simply be shared with all of the user's friends. This is likely to not only violate the user's privacy, but will also end up spamming the user's friends with irrelevant information. Facebook's failed experiment with frictionless sharing [5] is evidence of this. Therefore, *PullRec* instead enables any user to issue queries for entities about which they seek their friends' opinions. Here, we discuss how queries are processed in *PullRec*.

5.1 Query processing

Queries to *PullRec* are represented in the form of a three-tuple *(entity-type, topic, [location])*. The *entity-type* parameter specifies the type of entity for which the user is seeking recommendations, e.g., song, book, restaurant. The *topic* and *location* parameters serve as criteria that limit the subset of entities of interest. The *topic* argument is represented with a set of keywords, whereas the optional *location* argument is used as an additional criterion for real-world entities.

While processing any query from a user $Alice$, three properties need to be satisfied. First, to protect the privacy of $Alice$'s friends, if any of them have any information that is of relevance to $Alice$'s query, *PullRec* should share that information with $Alice$ only if authorized by that friend. Second, unless a user authorizes the sharing of her opinion about a particular entity with $Alice$, neither $Alice$ nor the OSN should be able to determine whether that user has *any* information that is relevant to $Alice$'s query. Finally, to protect $Alice$'s privacy and to ensure that *PullRec* does not spam her friends with irrelevant queries, we need to ensure that, in response to $Alice$'s query, only those friends of $Alice$ are prompted who have information relevant to $Alice$'s query.

To satisfy these properties, *PullRec* follows the following sequence of steps to serve a query issued by any user $Alice$. First, the OSN transmits $Alice$'s query to the *POAR* of each of her friends. Along with the query, the OSN sends along an OAuth token that lets $Alice$'s friends' *POARs* verify that the query is indeed from $Alice$; this helps a user's *POAR* ensure that its resources are utilized only for processing queries issued by the user's friends.

Second, at any friend's *POAR*, $Alice$'s query is matched against the information logged by *PullRec* for that user. If a user's *POAR* finds that it has no information that is of relevance to $Alice$'s query, then it does not respond to the OSN. If the user's *POAR* does indicate that its owner "consumed" an entity relevant to the query, two cases arise. On one hand, the user's *POAR* may have been able to successfully infer the user's opinion about the matching entity (e.g., when a user repeatedly visits the same restaurant or when the user's

brainwaves consistently reflect a positive sentiment when reading a particular article). If so, the user's *POAR* prompts him to check if he approves sharing this opinion with $Alice$. If authorized by the user, his *POAR* sends the relevant information to the OSN, which in return relays it to $Alice$.

On the other hand, in the case where a user's *POAR* is unable to infer the user's opinion about the matching entity, the *POAR* can notify the user about the matching entity and ask him to both express his opinion and his privacy preference (i.e., whether he is okay sharing this particular opinion with $Alice$). The user is required to put in more effort in this case because he needs to explicitly express his opinion. Therefore, to reduce overhead, a user can optionally configure his *POAR* to not prompt him in the latter case (i.e., when the user's *POAR* is unable to infer his opinion).

When a user's *POAR* finds information that is relevant to a user's query, it is important that the *POAR* send a notification to the user via a channel that is outside the purview of the OSN, e.g., via email or SMS. This is necessary to ensure that, when the OSN receives no reply from a user's *POAR* in response to a query, the OSN cannot determine whether that *POAR* contained any information relevant to the query.

5.2 Privacy-preserving enhancements

Two limitations remain in our recommendation sharing framework as described above. First, when a user's opinion is shared with her friend, *PullRec* reveals her friend's query to the user. This hurts the privacy of opinion seekers in comparison with existing *push*-based recommendation sharing platforms, wherein once a user shares her opinion with her friends, she is typically unaware which of them finds her opinion to be of interest. Second, push-based opinion sharing offers the possibility of a user serendipitously discovering a recommendation that she was not explicitly seeking. *PullRec*'s query-based sharing eliminates such information discovery.

To address these limitations, we rely on the fact that information sharing on most OSNs often happens at the granularity of *groups of friends*, rather than at the granularity of individual users. For example, users on Google+ share information with "circles" of friends, and Facebook users share using "friend lists". The effectiveness of these platforms indicate that users can and do express their privacy preferences at the granularity of groups of friends.

We leverage this information to modify recommendation sharing in *PullRec* as follows. First, when the OSN relays a user $Alice$'s query to the *POAR* of one of her friends, Bob, the OSN can include an OAuth token corresponding to that circle/friend list created by Bob which includes $Alice$; this is instead of including an OAuth token for $Alice$ as previously discussed. Thus, the OSN effectively anonymizes $Alice$'s query to an extent, because Bob may be unable to determine which of his friends in the specified circle issued the query. Note that this privacy is not perfect since Bob may be able to infer the query as being from $Alice$ based on the contents of the query and other contextual information.

Second, if Bob authorizes the sharing of his opinion with the circle associated with the query, the OSN can not only divulge Bob's opinion to $Alice$ but can also do so with other friends of Bob who are in that circle. Thus, other friends of Bob who are in the same circle as $Alice$ potentially benefit by discovering Bob's recommendation for an entity, about which they perhaps were not actively seeking an opinion. The OSN can avoid spamming Bob's friends by sharing this recommendation with those users who may find the information relevant; today, most OSNs have developed algorithms to infer their user's interests (e.g., Facebook's algorithm for determining what to show in a user's news feed).

6 Discussion

Incentive model. *PullRec*'s architecture requires every user to install monitoring software on her client devices and to pay for hosting a repository of all of the user's activities and opinions in the cloud. However, in return for the effort and cost borne by the user, she receives no benefit. The information stored in a user's *POAR* is beneficial only for the user's friends. Therefore, to offer users an incentive to opt in to *PullRec*, either a tit-for-tat incentive model will be necessary (wherein a user's *POAR* will respond to a friend's query only if that friend has opted in to *PullRec*), or the user must be able to derive some benefit from the information logged in her *POAR* (e.g., to serve her own queries [10]).

Aiding law enforcement. A user's *POAR* contains a significant amount of private information about the user, much of which is stored only ephemerally on the user's machine today or stored across several disparate providers. Thus, a user's *POAR* presents law enforcement with a rich log of a user's history. In our architecture of *PullRec*, information stored in a user's *POAR* is not accessible to anyone but the user without the user's explicit consent. However, the legal implications will need to be explored as to whether a user can resist law enforcement's request for information from her *POAR*.

7 Conclusions

Recommendation sharing between social contacts is an important utility that we are increasingly coming to rely upon. However, existing mechanisms for sharing recommendations impose significant overheads. Due to these overheads, we find that many users simply choose to not share entities (e.g., URLs and apps) that they like. We propose the *PullRec* framework to ensure that whenever a user is seeking a recommendation on a particular topic, she should be able to obtain one from a friend who has and is willing to share a relevant recommendation with the user. By carefully combining what information about a user is proactively logged and how this information is used to serve queries, *PullRec* moves us closer to comprehensive social sharing of recommendations.

8 References

[1] The 100 most influential news media Twitter accounts. http://memeburn.com/2010/09/the-100-most-influential-news-media-twitter-accounts/.

[2] After privacy fix, law firm's Netflix advocacy ends. http://legaltimes.typepad.com/blt/2013/04/after-privacy-fix-law-firms-netflix-advocacy-ends.html.

[3] Consumer reports: Half of social network users are oversharing, endangering privacy. http://readwrite.com/2010/05/04/consumer_reports_half_of_social_network_users_are_oversharing_endangering_privacy.

[4] Facebook developers - open graph. https://developers.facebook.com/docs/opengraph/.

[5] Frictionless sharing hits the skids at Facebook. http://www.theatlanticwire.com/technology/2012/09/facebook-realizes-nobody-wants-share-everything-all-time/57109/.

[6] Google+ platform - snippet. https://developers.google.com/+/web/snippet/.

[7] Neurosky MindWave. http://www.neurosky.com/Products/MindWave.aspx.

[8] Reverse geocoding. https://developers.google.com/maps/documentation/javascript/examples/geocoding-reverse.

[9] J. Allanson, T. Rodden, and J. Mariani. A toolkit for exploring electrophysiological human-computer interaction. In *INTERACT*, 1999.

[10] A. Balasubramanian, N. Balasubramanian, S. Huston, D. Metlzer, and D. Wetherall. FindAll: A local search engine for mobile phones. In *CoNEXT*, 2012.

[11] S. Brave and C. Nass. Emotion in human-computer interaction. *The human-computer interaction handbook: fundamentals, evolving technologies and emerging applications*, 2002.

[12] A. Machanavajjhala, A. Korolova, and A. D. Sarma. Personalized social recommendations: accurate or private. In *VLDB*, 2011.

[13] S. A. Paul, L. Hong, and E. H. Chi. Is Twitter a good place for asking questions? A characterization study. In *ICWSM*, 2011.

[14] M. Swan. Emerging patient-driven health care models: an examination of health social networks, consumer personalized medicine and quantified self-tracking. *International journal of environmental research and public health*, 2009.