Ask, Don’t Search: A Social Help Engine for Online Social Network Mobile Users

Tam Vu, Akash Baid
WINLAB, Rutgers University
{*tamvu,baid*}@*winlab.rutgers.edu*

***Abstract*—In this paper, we present the architectural details of Odin Help Engine, a novel social search engine that leverages online social networks and sensing data from mobile devices to find targeted answers for subjective queries and recommendation requests. In Odin, users’ queries are routed to those connections in their social network who (i) are most likely to help answer the question and (ii) can do so with a high level of confidence. Specifically, we first apply a link-based latent variable model to infer social relationships between users from their social network data to form a strength-weighted relationship graph. We then infer users’ expertise by context mining from social network data as well as from their mobile device sensor data. Lastly we apply pagerank-like algorithm that takes both relationship strength and user expertise into account to find a person that is most likely willing to answer the question posted by the user. We present the general design of the architecture and outline a location-related query example for detailed illustration.**

I. Introduction

Over the last decade, search engines have rapidly evolved and are today, perhaps the most useful resource that helps people get answers to their queries. Even though people could (and do) use search engines to find answers to most questions, today’s search engines are still far from an ideal solution. In particular, search engines are not good at abstract, subjective and recommendation type questions. For example, queries like: “What would be a good course from Rutgers’ Computer Science department next fall that is aligned with my research interests in machine learning and computer networks?” cannot be answered by regular search engines. In fact, it cannot even be expressed in a way that today’s search engines can understand, let alone answer it. The fundamental lacuna arises from the fact that search engines rely on links and content that *already* exists somewhere on the Internet and are thus inept at handling user-specific and contextual queries. They also fail to provide quality assurance, accountability and follow-up questioning that are usually associated with human to human interaction during the question-answer process. Coming back to the aforementioned example, a good way to get a useful answer for that question would be by posing it to all your friends or colleagues with domain expertise in computer science. However, querying all the people you know is very expensive in terms of time and energy and is thus not a viable solution. Instead, there should be a mechanism or a system to help find users the right person to ask such a question. That system and mechanism is the new type of question answering solution that is referred to as *Help Engine* in this paper.

The rapid growth of online social networks such as Facebook [1], LinkedIn [2], Twitter [3] provides a unique opportunity for such a class of help engines. Connections between users in such services can serve as links along which questions can be routed. Along the same lines, Aardvark [4], the state-of-the-art social search engine, was recently proposed to match questions from a user to other users based on their area of expertise. However, it requires participants to explicitly list their skill set. We argue that the assumption of users being fully aware of the topics that they can potentially answer about is too restrictive. For example, regular driving through a certain area would not be listed as an area of expertise, yet this is precisely the knowledge required to answer a question about the state of road-traffic in that area. A help engine which is capable of *finding out* its users’ domains of expertise would be much more powerful. In addition, Aardvark underutilizes the large amount of available data from social networks. Specifically, they simply use relationship information in the profile posted on those social network *without* considering latent relationships that can be inferred from activities between participants as well as similarities between them.

Also central to our architecture is the use of sensor data from mobile devices. The surge in smart phone penetration along with the abundance of sensors that are integrated into today’s mobile devices provides a great source of information that can be tapped by social search engines. However, to the best of our knowledge, there are no such systems that attempt to harvest mobile sensing information for use within social search engines. In this paper, we propose a help engine architecture to harvest sensing information collected by mobile devices and combine it with social relationships inferred from online social networks.

Contributions: Our contributions in this work are three-fold. We propose a system that applies machine learning techniques to infer relationships between participants and builds a weighted relationship graph among users. We harvest and index sensing information to build an expertise database upon which our pagerank-like algorithm is applied to find the best candidate that can answer a given question. We present a case study of location-related queries to illustrate the operation of our help engine.

This paper is organized as follows: the overview of our system is presented in Section II, followed by descriptions of each component,Sections III, IV and V. We illustrate the system operations by an example in Section VI. Section VII concludes the paper and points out future work directions.

II. System Overview

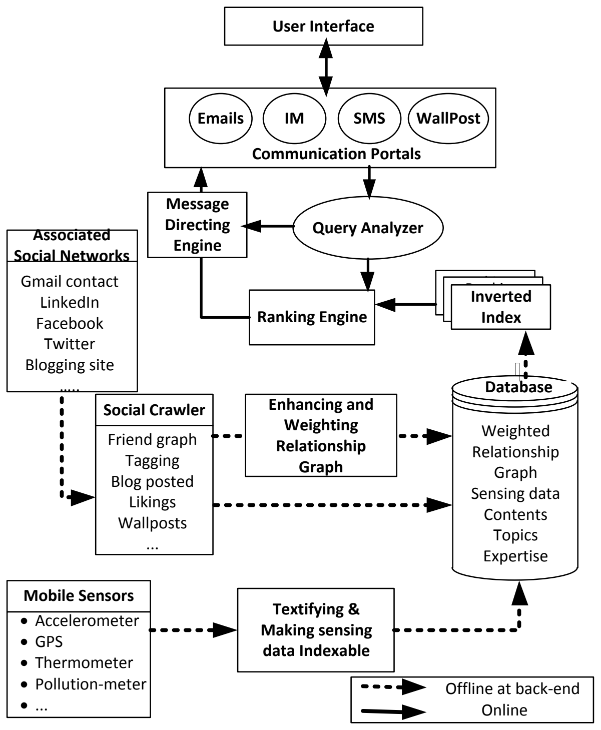

Fig. 1. Odin architecture overview

Our general approach is to build Odin with the help of four functional processes, of which three are done offline: (1) Mining social network profiles, joint activities between users and their photo/post tagging behaviour to create a strength-weighted relationship graph (WRG) through the use of machine learning techniques; (2) Crawling and indexing all the available resources on social networks to extract expertise information and creating a baseline indexed database (BiDB); (3) Converting sensor data and associated metadata to text in order to make it indexable and combining it with BiDB to create the indexed database (iDB); (4) Identifying and routing the query to the most suited responder by ranking users based on their relationship with the asker as well as their expertise. Before going into further details about individual components, we first describe the process of user registration and outline the query life assuming the existence of the indexed database and the weighted relationship graph.

A. User Registration

When a new user registers for Odin, our system initiates a number of steps to collect the data required for expertise indexing as well as relationship graph expansion. The first step involves getting social contacts from the user. In particular, the new user is requested to provide the credentials for his/her online social network accounts such as Facebook, LinkedIn and Twitter, from which friendship, affiliation information, photo/post tagging, and other activities are extracted. A user's relationship graph can be further expanded by sending out manual invitations to join Odin. In the next step, the user is asked to specify the types of sensing information he/she is willing to provide to the system. Optionally, the user is allowed to create and manage different groups of friends with different access rights to their sensing information. As an example, Bob can allow only the contacts in his *close friends* group to utilize his location information while allowing *everyone* in the network to use the data collected by the pollution sensors on his mobile device. As the last step for registration, the new user is asked to optionally provide or select the topics that he is willing or not willing to answer questions about. Once the registration process is completed, the user is active on the system and can readily ask and answer questions from other Odin members or anonymous users. Note that the indexing and relationship graph building processes run concurrently on the back-end while the user completes the registration (see Section III and Section V for the detailed description).

B. Query life

As a registered member, the user can start asking questions by publicly posting them on Odin's wall using a multitude of web-based interfaces, Odin UI on his own device, or by using third party plug-ins (e.g. Thunderbird plug-in, Facebook app, etc). The user can also classify the questions as private and restrict the list of of people who can view and answer the question. Figure 1 illustrates the architecture and flow of information in Odin. The query is first verified and analyzed by the *Query Analyzer* to determine the appropriate topics for the question. The analyzed question is then routed to the *Ranking Engine*, where the potential responders are selected using a ranking algorithm similar to those used in corpus-based search engines. In particular, the Ranking Engine accesses the Inverted Index and the Weighted Relationship Graph (WRG) for the list of candidate responders, and ranks them to identify the ones that have the most likelihood of willingness to answer the question with the highest level of confidence. Restrictions specified in the query are taken into account by the ranking engine to eliminate responder candidates that violate the restrictions, regardless of their ranking. Once the top candidates are found, the Ranking Engine simultaneously or sequentially forwards the query to them to get the answer. The answers are then routed from the responder to the asker via the Message Direction Engine. If the asker is not completely satisfied with the answer, he/she can ask followup questions to improve the satisfaction. In all cases, the asker has an option of leaving a feedback on how well the question was answered to help Odin improve its routing performance.

III. Intimacy inference for Weighted Relationship Graph

The friendship connectivity graph directly extracted from user profiles on online social networks (e.g. friend list on Facebook, contact list on Google mail, or connection list on LinkedIn) is not sufficient for grading relationships. It merely indicates binary relational ties between users (i.e. are they connected or not). Evidently, binary relationship graphs do

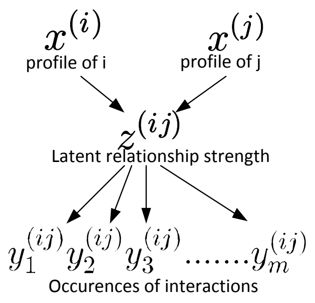
$x^{(i)}$
profile of i
$x^{(j)}$
profile of j
$z^{(ij)}$
Latent relationship strength
$y_1^{(ij)} y_2^{(ij)} y_3^{(ij)} \ldots\ldots y_m^{(ij)}$
Occurences of interactions

Fig. 2. Relationship strength model illustration

not capture the intimacy level between users, which plays a crucial role in predicting the willingness to answer a question from the user. Hence, inferring the intimacy, or strength, of the relationship is important.

Based on habitual similarities and interactions between users, Xiang et al. [5] proposed a latent variable model (LVM), to distinguish strong relationships from the weak ones. Similar latent factor analysis has been used in other machine learning algorithms, for example in the winning entry of the Netflix prize [6]. Leveraging this technique, we prune and augment the original binary connectivity graph into a relationship graph where links between users are weighted by the strength of their connection. LVM relies on two assumptions about the correlation of 'strength of the relationship' with profile similarity and frequency of interaction. Homophily theory [7] from sociology postulates that people with similar characteristics tend to form ties with one another and vice versa, which means the stronger the ties, the higher the similarity [8]. Further more, closer the relationship, the more frequent interaction between individuals happens. Interactions in the realm of online social network range from profile viewing, picture and comment tagging to private messaging. Since everyone lives on a limited amount of time and energy resources, one tends to direct those resources towards the relationships that have higher priority, and are more important. Based on these two assumptions, the relationship strength is modeled as the *hidden cause* of user interactions while profile similarity serves as prior knowledge to the model.

We use the latent variable model described in [5] in the manner described below. In order to rigorously define the model, we use the following notations: let $x^{(i)}$ and $x^{(j)}$ be the profile vectors of two individuals i and j, where each element of these vector could denote, for example the individual's affiliation, place of residence, hobbies, etc. Let $y_t^{(ij)}$ for $t = 1, 2, \ldots, m$ be the occurrences of m unique interactions between the two individuals. Further, let $z^{(ij)}$ denote the *latent (or hidden) relationship strength* between i and j: higher this value, more 'stronger' is the relationship between i and j. The key purpose of LVM is to infer $z^{(ij)}$ based on $x^{(i)}$ and $x^{(j)}$. It also predicts the influence of $z^{(ij)}$ on $y_t^{(ij)}$. The relationship between these variables can be illustrated by a directed graph as shown in Figure 2. The model can be considered as a combination of discriminative ($p(Z|X)$) and generative ($p(Y|Z)$) components corresponding to the upper and the lower parts of the graph respectively. Thus the likely causal relationship among those variables can be decomposed as:

$$\begin{aligned} &P(z^{(ij)}, \boldsymbol{y^{(ij)}} | \boldsymbol{x^{(i)}}, \boldsymbol{x^{(j)}}) \\ &= P(z^{(ij)} | \boldsymbol{x^{(i)}}, \boldsymbol{x^{(j)}}) \prod_{t=1}^{m} P(y_t^{(ij)} | z^{(ij)}) \end{aligned} \quad (1)$$

As shown in [5], although the latent strength value z captures the similarities and interactions between a pair of users, it is difficult to directly compute from online user data. Instead, it can be estimated for each pair of people to maximize the overall observed data likelihood. It is worthy to note here that the model can be applied to estimate the strength of directed as well as undirected relationships. This asymmetric attribute of the LVM model is indeed desirable, since it reflects real world social connections (e.g. one can consider another person his/her best friend but not vice versa. Similarly, one can tag a photo of the other person but not the other way around). For our social help engine, we assume the widely used Gaussian distribution to model the conditional probabilities $P(z^{(ij)} | \boldsymbol{x^{(i)}}, \boldsymbol{x^{(j)}})$. Then the conditional probability between $z^{(ij)}$ and $\boldsymbol{x^{(i)}}, \boldsymbol{x^{(j)}}$ is as follows:

$$P(z^{(ij)} | \boldsymbol{x^{(i)}}, \boldsymbol{x^{(j)}}) = \mathcal{N}(\boldsymbol{w^T s^{(ij)}}, v) \quad (2)$$

where $\boldsymbol{s^{(ij)}}$ is a similarity vector computable from $x^{(i)}$and $x^{(j)}$, $\boldsymbol{w}$ is a n-dimensional weight vector to be estimated and v is the variance in Gaussian model which is manually configured.

From (1), we need to model the conditional probability of $y_t^{(ij)}$ given $z^{(ij)}$. To avoid including interactions which are independent of user relationships, a set of auxiliary variables $a_{t1}^{ij}, a_{t2}^{ij}, \ldots, a_{tq}^{ij}$ are introduced. The logistic function to model $y_t^{(ij)}$ given $z^{(ij)}$ and a_t^{ij} can be written as:

$$P(y_t^{ij} = 1 | z^{(ij)}, a_t^{ij}) = \frac{1}{1 + e^{-(\boldsymbol{\theta_t^T u_i^{(ij)}} + b)}} \quad (3)$$

where $\boldsymbol{u_i^{ij}} = [z^{(ij)}, a_t^{ij}]^T$ and $\boldsymbol{\theta_t} = [\theta_{t1}, \theta_{t2}, ..., \theta_{t(q+1)}]^T$. Additionally, $L2$ regularizers are applied on the parameters $\boldsymbol{w}$ and θ to avoid over-fitting and finally we get the computable equation for the joint probability as Eq.(7) in [5]. Once the joint probability is found, we can estimate the latent variable z by treating it as a parameter and finding a set of point estimates $\hat{\boldsymbol{W}}, \hat{\boldsymbol{\theta}}, \hat{\boldsymbol{z}}$ that maximizes the likelihood $P(\boldsymbol{y}, \hat{\boldsymbol{W}}, \hat{\boldsymbol{\theta}}, \hat{\boldsymbol{z}} | \boldsymbol{x})$. A coordinate ascent optimization scheme is applied to find $z^{(ij)}$ by using Newton-Raphson iterative approach by utilizing Eqs.(11)-(16) in [5].

We use the approach described above to find the strength of the relationship between all users in Odin, which is then utilized to convert the original connectivity graph into a weighted relationship graph. We note that although the bootstrapping for estimating the latent strength values has a computational complexity of $O(N^2)$ for N users, it only needs to be done in the offline mode and thus does not add delays to the query-response process.

IV. Data Collection

A. Device signal harvesting

The Odin help engine takes advantage of readily available sensory data from mobile devices to enrich the expertise database and to infer user context. Sensor-equipped devices such as mobile phones, tablets and laptops that are registered to Odin users can report their raw sensor readings attached with a timestamp to the Odin server. The server combines these raw data streams with additional application-specific databases (ASD) to add meaning and semantics into the data before dumping it to the iDB for indexing. We illustrate the process by taking location sensing information as an example. A raw time series of <latitude, longitude> tuples stored in the database does not benefit Odin directly. For it to be better utilized, it must be converted into text-form indexable information. For example, it can be converted to a series of street addresses by using the Google reverse geocoding service [9]. Indeed, this time-stamped street address information coupled with other sensing data such as movement speed can help answer the class of questions on real-time traffic monitoring. Further, such information combined with accelerometer traces can help answer questions related to road quality on specific streets utilizing techniques outlined in [10]. In general, harvesting of sensor signals requires a dedicated database and related transformation software that can convert the raw sensing information into human readable and machine indexable information.

B. Social Crawling for Fine Tuning of Expertise Database

In addition to the mobile sensors, another important source of data is the user generated content on social networks. In Odin, identification of topics that a user might be able to answer is of utmost importance. Our current framework incorporates the following four mechanisms for extracting and indexing social network context. Additional mechanisms can be easily added to further fine tune the expertise database.

- *Blog posts extraction:* Users are most likely to blog about things they know and/or are interested in. Thus we enhance the user expertise scores based on keywords found in any linked blogs.
- *Online social network profile:* Professional networks like LinkedIn offer a good source of indexable expertise areas for Odin users. Data extracted from social networks are weighted based on the source: for example data from LinkedIn or Monster is given a higher weight compared to those from Facebook or Twitter and the relative weights can be further optimized for performance.
- *Online tagging and comments:* To parse comments and tags, we apply a linear SVM technique to first categorize the general topic of the tag and comments and then run the TF-IDF taxonomy-based extractors to identify expertise related keywords.
- *Satisfaction Feedback:* Last but not the least important source of information for fine tuning the expertise database is the feedback from other users. For example, if a prospective responder repeatedly declines to answer questions about a specific topic, the ranking of that user for that particular topic is reduced.

V. Ranking algorithm

Once the intimacy graph and the expertise database is available, the key online component required is the Ranking Engine. In this section, we describe the functioning of the Ranking Engine in detail, starting with a statistical model of the algorithm.

Simply ranking the users based on the relevance of their expertise to the question being asked is inefficient. For example, if the highest ranked user does not have any relationship with the asker, there is a high chance that he/she might decline to answer the question. Hence the goal of our ranking algorithm is to simultaneously maximize the probability of getting an answer back and the probability of the answer being of high quality. To that end, we use an adaptive form of the ranking algorithm proposed in [4] based on our Weighted Relationship Graph. Let T denote the set of all possible topics archived in Odin and $t \in T$ be a topic that a question q belongs to. The probability $p(u_i|q)$ that user i will successfully answer the question q depends on level of expertise of user u_i on the topic t. Statistically, it can be represented as:

$$p(u_i|q) = \sum_{t \in T} p(u_i|t)p(t|q) \tag{4}$$

Further, we define the query-independent probability $p(u_i, u_j)$ that user u_i is willing to answer a question from user u_j regardless of the question. Since we rely on the assumption that stronger the relationship between two users, higher is the probability they are willing to answer questions from one another, we can estimate $p(u_i, u_j)$ as:

$$p(u_i, u_j) = WRG(u_i, u_j) \tag{5}$$

Note that this optimization is in sharp contrast with the existing social search engine [4] where no data about the strength of connection is available. We define $r(u_i, u_j, q)$, the scoring function for question q for the user pair i, j as the composition of the two probabilities:

$$\begin{aligned} r(u_i, u_j, q) &= p(u_i|u_j)p(u_i|q) \\ &= p(u_i|u_j) \sum_{t \in T} p(u_i|t)p(t|q) \\ &= WRG(u_i, u_j) \sum_{t \in T} p(u_i|t)p(t|q) \end{aligned} \tag{6}$$

In equation (6), $WRG(u_i, u_j)$ and $p(u_i|t)$ are pre-computed upon user signups and is continuously updated in the background through techniques described in Section IV-A. Categorizing the question, which is reflected in the $p(t|q)$ term is the only computation which is done online, after a question is asked. Similar to legacy search engines like Google and newer social search engines [4], we use probabilistic classification [11] to derive the set of topics given a certain query term q.

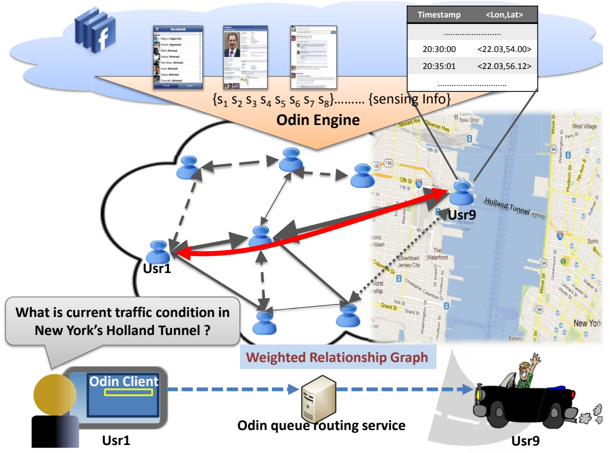
Timestamp <Lon,Lat>
20:30:00 <22.03,54.00>
20:35:01 <22.03,56.12>
{s1 s2 s3 s4 s5 s6 s7 s8}......... {sensing Info}
Odin Engine
Usr9
Holland Tunnel
Usr1
What is current traffic condition in New York's Holland Tunnel ?
Weighted Relationship Graph
Odin Client
Odin queue routing service
Usr1
Usr9

Fig. 3. Example of location-based query: The Odin Engine assimilates sensor data, profiles, friend list, wall posts, etc to compute the Weighted Relationship Graph. Using this information, traffic related query from User 1 is routed to User 9. The Odin client of User 9 can automatically respond to the query by parsing the location data available at the server

VI. Use case analysis for location based queries

In this section, we present a use case example of Odin being utilized for location based queries. In order to build the WRG graph, we first need to identify the list of similarities and interactions which would serve as inputs for the LVM algorithm. We define similarities between user i and user j as $s^{(ij)} = [s_1^{(ij)}, s_2^{(ij)}, \ldots, s_8^{(ij)}]$ with the first six variable denoting similarities in the following attributes: school, company, geographical region, industry, job title, functional area; and $s_7^{(ij)}$, $s_8^{(ij)}$ being the logarithm of the normalized counts of common groups and connections respectively between users i and j. Here $s_l^{(ij)}$ for $l \in [1, 6]$ assumes the value 1 when i and j share the same attribute and 0 otherwise. Further, we define four interactions $y^{(ij)} = [y_1^{(ij)}, y_2^{(ij)}, y_3^{(ij)}, y_4^{(ij)}]$ between users representing wall posts, recommendations, profile views and picture tags. The $y_t^{(ij)}$ variables assume binary values depending on the interactions between i and j. For example, if user i has written a recommendation for user j, then $y_2^{(ij)} = 1$ and 0 otherwise. For each pair of users, Odin uses the algorithm outlined in section III to infer the WRG graph. In order to augment the system with location data, GPS traces in the form of <latitude, longitude> tuples are collected from users and converted to street name data using the Goolge reverse geo-coding service [9]. Online profile information, tagging activities, wall posts and the converted location information is stored in the indexed database. The expertise database is next synthesized using the procedures outlined in Section IV-B.

The Odin Engine configured with the set of variables defined above can be readily used for location specific queries such as: "What is the current traffic condition at Holland Tunnel, NY ?" Figure 3 illustrates the routing of such a query from User 1. After being verified by the Query Analyzer and categorized as a location-related query, it is forwarded to the Ranking Engine. Here the iDB is invoked to find if any other users with strong relationship with User 1 has data that matches the query content. In the example above, User 9 happens to have passed through the same location and thus the query is answered using User 9's database records.

VII. Conclusion and future work

In this paper, we presented the architecture of Odin, a social search engine that leverages data streams from online social networks and mobile devices to help users find answers to contextual and subjective questions. Through Odin, user queries are routed to a subset of their social contacts who are most likely to answer the questions and whose expertise match with the query content. The system makes use of a link-based latent variable model to infer social relationships between users from their social network profiles to form a strength-weighted relationship graph. User expertise is determined by mining social network data as well as sensor data from mobile devices. Lastly a pagerank-like algorithm that takes both the relationship strength and the user expertise into account is applied to find the right set of users to forward the query.

We outlined the details of each component required to realize such a system and are currently working on implementing it as an online service. We believe that the novel combination of link-based latent variable model and automatic classification of user expertise offers a powerful mechanism for realization of more useful search engines. The basic framework presented here can be augmented with a number of features to make it more useful and deployable.

On-going work targets a number of system components including: (i) intelligent sampling and data compression mechanisms to address energy constraints in mobile devices; (ii) signal fusion from multiple sensors and from different sets of social network data; (iii) incentive mechanisms and business model to encourage participation. Future work also involves rigorous mechanisms to ensure the privacy of sensitive information which is critical for the successful deployment of such a system.

References

[1] "Facebook - Online social network." [Online]. Available: http://www.facebook.com/
[2] "LinkedIn - Online professional social network." [Online]. Available: http://www.linkedin.com/
[3] "Twitter - Follow your interests." [Online]. Available: http://www.twitter.com/
[4] D. Horowitz and S. D. Kamvar, "The Anatomy of a Large-Scale Social Search Engine," *WWW 2010, April 26-30, 2010,Raleigh, North Carolina*.
[5] R. Xiang, J. Neville, and M. Rogati, "Modeling relationship strength in online social networks," *WWW '10*, p. 981, 2010.
[6] J. S. Breese, D. Heckerman, and C. M. Kadie, "Empirical analysis of predictive algorithms for collaborative filtering," in *UAI*, 1998, pp. 43–52.
[7] M. McPherson, L. Smith-Lovin, and J. M. Cook, "Birds of a Feather: Homophily in Social Networks," *Annual Review of Sociology*, vol. 27, no. 1, pp. 415–444, Aug. 2001.
[8] M. Granovetter, "The Strength of Weak Ties: A Network Theory Revisited," *Sociological Theory*, vol. 1, p. 201, 1983.
[9] "Google Geo Developers Blog: Geocoding... in Reverse!" [Online]. Available: http://googlegeodevelopers.blogspot.com/2008/10/geocoding-in-reverse.html
[10] J. Eriksson, L. Girod, B. Hull, R. Newton, S. Madden, and H. Balakrishnan, "The pothole patrol: using a mobile sensor network for road surface monitoring," in *Proceedings of MobiSys, 2008*, pp. 29–39.
[11] M. Jaeger, "Probabilistic classifiers and the concepts they recognize." The AAAI Press, 2003, pp. 266–273.