Money for Nothing and Privacy for Free?

Swapneel Sheth, Tal Malkin, Gail Kaiser
Department of Computer Science, Columbia University, New York, NY 10027
{swapneel, tal, kaiser}@cs.columbia.edu

ABSTRACT

Privacy in the context of ubiquitous social computing systems has become a major concern for society at large. As the number of online social computing systems that collect user data grows, concerns with privacy are further exacerbated. Examples of such online systems include social networks, recommender systems, and so on. Approaches to addressing these privacy concerns typically require substantial extra computational resources, which might be beneficial where privacy is concerned, but may have significant negative impact with respect to Green Computing and sustainability, another major societal concern. Spending more computation time results in spending more energy and other resources that make the software system less sustainable. Ideally, what we would like are techniques for designing software systems that address these privacy concerns but which are also sustainable — systems where privacy could be achieved "for free," *i.e.,* without having to spend extra computational effort. In this paper, we describe how privacy can indeed be achieved for free — an accidental and beneficial side effect of doing some existing computation — in web applications and online systems that have access to user data. We show the feasibility, sustainability, and utility of our approach and what types of privacy threats it can mitigate.

Categories and Subject Descriptors

K.4.1 [**Computing Milieux**]: Public Policy Issues—*Privacy*; H.3.3 [**Information Systems**]: Information Search and Retrieval—*Information Filtering*

General Terms

Algorithms, Human Factors

Keywords

Social Computing, Web 2.0, Correlation Privacy, Concept Drift, Differential Privacy

1. INTRODUCTION

Today's college students do not remember when social recommendations, such as those provided by Amazon, Netflix, Last.fm, and StumbleUpon, were not commonplace. Privacy in the context of these social computing systems has become a major concern for the society at large. A search for the pair of terms "facebook" and "privacy" gives nearly two billion hits on popular search engines. Recent feature enhancements and policy changes in social networking and recommender applications — as well as their increasingly common use — have exacerbated this issue [11, 26, 30, 58]. There has been a lot of media attention on privacy — *e.g.,* the AOL anonymity-breaking incident reported by the New York Times [4]). Both users of the systems and even non-users of the systems (*e.g.,* friends, family, co-workers, etc. mentioned or photographed by users) are growing more and more concerned about their personal privacy [44].

Social computing systems, when treated in combination, have created a threat that we call "Correlation Privacy." Narayanan and Shmatikov [40] demonstrated a relatively straightforward method to breach privacy and identify individuals by correlating anonymized Netflix movie rating data with public IMDb data. A similar de-anonymization approach could potentially be applied to any combination of such data-gathering systems, so how to safeguard again these "attacks" is an important concern for the designers of social computing systems. This is analogous to earlier work addressing queries on census data but, at that time, there were relatively few prospective attackers [1, 5].

There has been a lot of recent work on data anonymization for privacy [2, 15, 24, 27, 34, 42, 48, 51]. However, data anonymization alone may not be sufficient as Narayanan and Shmatikov show. For more details on the related work including de-anonymization approaches, please see Section 6. We need other techniques to deal with privacy concerns, which may be used orthogonal to data anonymization.

In this paper, we propose an approach, which we call "Privacy for Free," targeted towards online social systems. In particular, we focus on systems that already have access to user data such as purchase history, movie ratings, music preferences, and friends and groups and that use complex data mining techniques for providing additional social benefits such as recommendations, top-n statistics, and so on to their users. The problem we deal with is users who have intentionally disclosed data on a public system, entering their data via web browsers onto some website server that is known to make publicly available certain data-mined community knowledge gleaned from aggregating that data

with other users — but the users don't want their data to be personally identifiable from the aggregate.

The main research question we try to answer here is — Is there an approach that can be used with complex web applications and software systems, that will achieve privacy without spending any extra resources on computational overhead? We believe it is — our key insight is that we can achieve privacy as an accidental and beneficial side effect of doing already existing computation.

The already existing computation in our case is weighing user data in a certain way — weighing recent user data exponentially more than older data to address the problem of "concept drift" [54] — to increase the relevance of the recommendations or data mining. This weighing is very common and used in a lot of systems [18, 29, 32, 38]. Recent work in the databases/cs theory communities on Differential Privacy [20, 36] led to our insight that our already existing computation for weighing user data is very similar to one of the techniques for achieving differential privacy. Intuitively, differential privacy ensures that a user's participation (versus not participating) in a database doesn't affect his privacy significantly. We provide more detailed information on Differential Privacy in Section 3. This resulted in the formulation of our hypothesis — if we change the concept drift computation so it matches the technique for achieving differential privacy (which would be a very minor and straightforward code change as the two techniques are very similar), would we get privacy as a beneficial side effect of addressing a completely different problem?

We show that it is indeed possible to get privacy as a beneficial side effect of addressing concept drift — thus, privacy for free — and this is the main contribution of our paper. Our approach can be used in certain social computing systems and web applications to achieve "privacy for free," and we show the feasibility, sustainability, and utility of using this approach to building software systems. We also contribute to the discussion in the privacy community about how to define privacy and how to achieve it. Specifically, we suggest a new direction for designing (differentially, or otherwise) private algorithms and systems motivated by using the beneficial side-effects of doing some already existing computation.

The rest of the paper is organized as follows: Section 2 describes the motivation of our problem and why making privacy sustainable is important. Section 3 provides background information on Differential Privacy and Concept Drift. Section 4 describes our "Privacy for Free" approach. Section 5 presents our empirical evaluations to show the feasibility, sustainability, and utility of our approach. In Section 6, we describe the related work and finally, we end the paper in Sections 7 and 8 with some discussion about our results and our conclusions.

2. MOTIVATION

Green Computing (or Green IT) is "the study and practice of designing, manufacturing, using, and disposing of computers, servers, and associated subsystems [...] efficiently and effectively with minimal or no impact on the environment" [39]. With our oil reserves projected to exhaust in less than fifty years [52], and renewable energy sources still providing only a small fraction [50], Green Computing here and now is becoming more and more important and, indeed, vital to our children and grandchildren.

An important research direction will be investigating how to build greener and more sustainable social computing systems and web applications, in addition to the complementary algorithmic efficiency and systems perspective such as resource allocation, platform virtualization, and power management pursued by other computer science subdisciplines [37]. Ideally, from a sustainable software system point of view, we want to build systems that solve real-world problems by spending very little (or no) extra computational effort.

There has been a lot of recent work in the research community on data privacy [2, 15, 24, 27, 34, 42, 48, 51]. This has focused on anonymization techniques to hide sensitive data. For example, Clause and Orso [15] propose techniques for automated anonymization of field data for software testing; Grechanik *et al.* [27] and Taneja *et al.* [48] propose using k-anonymity [47] for privacy by selectively anonymizing certain attributes of a database for software testing. These approaches require additional CPU resources to guarantee privacy. Our work is orthogonal — we try to answer the question as to whether it's possible to get privacy without requiring any additional CPU overhead.

This is our main motivation for this paper. We feel that our "Privacy for Free" approach can result in social computing systems that are more sustainable and that have privacy guarantees built in.

3. BACKGROUND

Here we provide some background information on Differential Privacy and Concept Drift.

3.1 Differential Privacy

In the 1970s, when research into statistical databases was popular, Dalenius [17] proposed a desideratum for statistical database privacy — access to a statistical database should not enable someone to learn something about an individual that cannot be learned without access to the database. While such a desideratum would be great for privacy, Dwork *et al.* [20, 21] showed that this notion of absolute privacy is impossible using a strong mathematical proof. The problem with the desideratum is the presence of "Auxiliary Information". Auxiliary Information is similar to, and a generalization of, the notion of Correlation Privacy mentioned earlier.

Dwork gives a nice example to explain how Auxiliary Information can be a problem when privacy is concerned — "Suppose one's exact height were considered a highly sensitive piece of information, and that revealing the exact height of an individual were a privacy breach. Assume that the database yields the average heights of women of different nationalities. An adversary who has access to the statistical database and the auxiliary information "Terry Gross is two inches shorter than the average Lithuanian woman" learns Terry Gross' height, while anyone learning only the auxiliary information, without access to the average heights, learns relatively little." An interesting observation made by Dwork is that the above example for breach of privacy holds regardless of whether Terry Gross' information is part of the database or not.

To combat Auxiliary Information, Dwork proposes a new notion of privacy called Differential Privacy. Dwork's paper is a culmination of the work started earlier and described in papers such as [9, 19, 23]. Intuitively, Differential Privacy guarantees privacy by saying that if an individual partici-

pates in the database, there is no additional loss of privacy (within a small factor) versus if he had not participated in the database. Formally, Differential Privacy is defined as follows: A Randomized function K gives ϵ-differential privacy if for all data sets D_1 and D_2 differing on at most one element, and all $S \subseteq Range(K)$,

$$\Pr[K(D_1) \in S] \leq \exp(\epsilon) \times \Pr[K(D_2) \in S] \quad (1)$$

The notion of all data sets D_1 and D_2 captures the concept of an individual's information being present in the database or not. If the above equation holds, it implies that if an individual's information is present in the database, the breach of privacy will be almost the same if that individual's information was not present. Differential Privacy is now commonly used in the database, cryptography, and cs theory communities [10, 21, 22, 43].

We like the definition of Differential Privacy due to its strong mathematical foundations, which can allow us to prove/disprove things theoretically. From a software web application developer's point of view, they can tell their users — "Look, our system is differentially private. So if you decide to use our system and give it access to your data, you are not losing any additional privacy (within a small factor) versus if you did not use our system. In other words, the probability of bad things happening to you (in terms of privacy) is roughly the same whether you use our system or not."

3.2 Achieving Differential Privacy

Dwork describes a way of achieving differential privacy by adding random noise. In the Terry Gross height example above, instead of giving the true average, the system would output average$\pm\delta$, where δ would be randomly chosen from a mathematical distribution. Thus, the adversary wouldn't be able to find out the exact height of Terry Gross. Since then, there have been many papers that have proposed different mechanisms for achieving differential privacy [10, 21, 22, 43].

A mechanism of note for achieving differential privacy was proposed by McSherry and Talwar [36] called the "Exponential Mechanism" (EM). The EM algorithm is as follows: Given a set of inputs, and some scoring function that we are trying to maximize, the algorithm chooses a particular input to be included in the output with probability proportional to the exponential raised to the score of the input using a scoring function. Thus, inputs that have a high score from the scoring function have an exponentially higher probability of being included in the output than those inputs that have a low score. McSherry and Talwar prove that this EM algorithm is differentially private.

Consider the Terry Gross example from above and let's assume that the database has historical data going back 100 years. The average heights of people change over time so giving an average height over the 100 years is not very useful. If the scoring function we use is to maximize the recency of data, newer data elements will be chosen with exponentially higher probability that older data elements to be included in the average. Since we are doing this probabilistically, the exponential probability weighing ensures that the exact answer is not revealed and that differential privacy is maintained. This EM algorithm is one of the corner stones of our "Privacy for Free" approach and we describe how it's used in the next section.

3.3 Concept Drift

People's preferences change over time — things that I like doing today may not be things I liked doing 10 years ago. If data is being mined or recommendations being generated, the age of the data needs to be accounted for. To address this problem, the notion of Concept Drift was formed [54]. This problem needs to be addressed by any field that deals with data spanning some time frame (from a few hours to months and years). An example class of systems that need to address the problem on Concept Drift is Recommender Systems. Many recommender systems use Collaborative Filtering (CF), *i.e.,* recommending things to an individual by looking at what other users similar to the individual like [28, 38, 55]. CF algorithms typically look at the activities of individuals from the past (movies watched, things bought, etc.) and use this to derive recommendations. However, people's preferences change over time. For example, when I am in college and taking a lot of classes, I might buy a lot of textbooks from Amazon. When I graduate, I may not need textbook recommendations. This is exactly the kind of problem that Concept Drift tries to address.

Other example classes of systems that need to address this problem are social software systems [6], systems for collaboration and awareness [53], systems that mine online software repositories [13], etc. For these kinds of systems, there is a lot of old and recent data available and weighing certain data differently might be essential.

3.4 Addressing Concept Drift

There have been many different solutions proposed to address the problem of Concept Drift [31, 33, 54]. A particular solution of note is the Exponential Time Decay Algorithm [16] (ETDA, henceforth). ETDA weighs things done recently exponentially higher than things done in the past. It gradually decays the weight of things done in the past so that things done in the distant past do not affect the outcome as much as things done recently, thus addressing the problem on Concept Drift.

$$g(x) = \exp(-l \times x), \text{ for some } l > 0 \quad (2)$$

The non-increasing decay function using by ETDA is shown in Equation (2). ETDA is very popular and used by a lot of systems [18, 29, 32, 38]. For the rest of the paper, we refer to this as the CD (Concept Drift) algorithm.

Consider the Terry Gross example again and let's assume that the database has historical data going back 100 years. As average heights change over time, the CD algorithm will weigh newer data exponentially higher than older data resulting in a weighted average height. This would reflect the recent trends but also account for older data. The CD algorithm is the another corner stone of our approach and we build on it more in the next section.

4. PRIVACY FOR FREE

The EM algorithm can use a variety of scoring functions — McSherry and Talwar show different scoring functions for privacy preserving auctions [36]. In such scenarios, the EM and CD algorithms are not similar. Using the timestamp scoring function is what makes them similar to each other. The CD algorithm uses exponential weighing over the data while the EM algorithm chooses inputs with probability proportional to the exponential of the scoring function.

```
public double getWeightedValue() {
  double value = 0;
  for(int i=0; i<array.length; i++) {
    double weight = Math.exp(-i);
    value += weight*array[i];
  }
  return value;
}
```

Listing 1: Java code for the CD algorithm

```
public double getWeightedValue() {
  double value = 0;
  for (int i = 0; i < array.length; i++) {
    double weight = Math.exp(-i)*0.5;
    double probability = Math.random();
    if (probability < weight) {
      value = array[i];
      return value;
    }
  }
  return value;
}
```

Listing 2: Java code for the EM algorithm

Only if we choose the scoring function for the EM algorithm to be the timestamp of the data, the two algorithms becomes similar. The CD algorithm is deterministic and weighs new data exponentially higher than older data; the EM algorithm is probabilistic and chooses new data with an exponentially higher probability than older data.

The Java code for the CD algorithm and the EM algorithm using the timestamp scoring function are shown in Listings 1 and 2 respectively. In terms of running time complexity, the CD algorithm is $O(n)$. For EM (using the timestamps scoring function), the worst case is also $O(n)$. However, as we use randomization, the expected running time is sublinear — $o(n)$.

This is the crux of our paper — if existing systems that already use the CD algorithm modify the code to use the EM algorithm instead, they would, as an added benefit, get the main advantage of the EM algorithm — differential privacy. Further, this privacy would not require any extra computational overhead and thus, we would get privacy for free.

Since these two algorithms are very similar, it would require a very small and straightforward change to the code to change from the CD algorithm to the EM algorithm. We would need to replace the CD code with the EM code shown above. This would be a one-time change and could be done by adding a new library method for EM or done statically via refactoring and could even be automated.

The important requirement for the differential privacy guarantees to hold are that all the data access must be done via the EM algorithm, which could be implemented as a separate class or be part of a library or the data model, etc.

5. EVALUATION

Our approach requires implementing (or substituting an existing implementation of the CD algorithm with) the EM algorithm. To evaluate our approach, we implemented the EM and CD algorithms and investigated the differences in these. Our goal was to answer the following research questions:

RQ1: Feasibility—Does using our approach guarantee differential privacy?

RQ2: Utility—Does using our approach affect the utility of the system to give meaningful recommendations or mine data?

RQ3: Sustainability—Can our approach be sustainable? Can using our approach result in no additional computational resources for privacy?

With RQ1, we aim to prove the primary benefit of our approach — guaranteeing privacy. Our goal is to show that it does indeed guarantee differential privacy making it suitable to be used in a variety of social systems and web applications.

With RQ2, we explore the utility of using our approach. A "straw man" way to guarantee privacy for any recommender/data mining system is to give a random answer every time. This would not require any clever technical solutions, but this would be very bad for the overall utility of the system — the goal of most such systems is to provide relevant information. There exists a tradeoff between accuracy and privacy and we explore this here. We aim to show that, using our technique, there is a small loss in accuracy and that this loss in accuracy scales very well (roughly constant) as the size of the system increases. Thus, if a small loss in accuracy is acceptable, we can get privacy for free without spending any additional computational resources.

With RQ3, we aim to show the sustainability benefits of using our approach. We show that using our approach (and the EM algorithm) requires less CPU time than the equivalent CD algorithm. Not only do we not need any additional computational resources, we should be able to reduce computational needs by using our approach.

5.1 RQ1 — Feasibility

Our approach requires the use of the EM algorithm for all access to the data. The EM algorithm that we require is exactly the same as the one proposed by McSherry and Talwar [36]. The algorithm they propose can work with different scoring functions that weigh the data differently — in our case, the scoring function we use is the timestamp of the data. Our use of the EM algorithm in our approach can thus be viewed as an instantiation of the general EM algorithm. McSherry and Talwar show a theoretical proof for the EM algorithm to be differentially private. We do not repeat the proof here and we encourage the interested reader to look at the paper (page 5 of [36]). As all data access happens via the EM algorithm, our approach also guarantees differential privacy.

5.2 Methodology

For RQ2 and RQ3, we carried out experiments to validate our hypotheses. We use synthetic data as there are no benefits of using real world data for our hypotheses. We create an array of size n and randomly fill it with values from 0 to

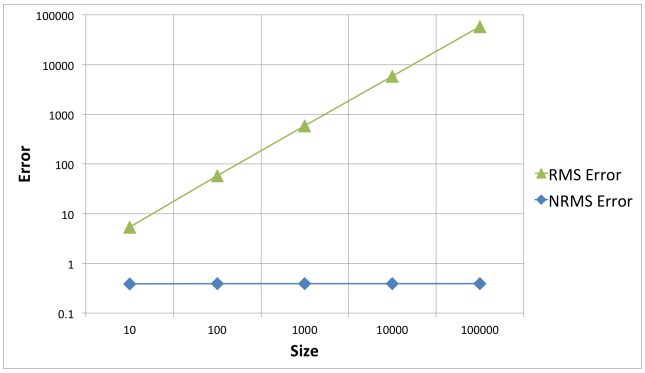

Figure 1: RMS and NRMS Error vs Size of data set

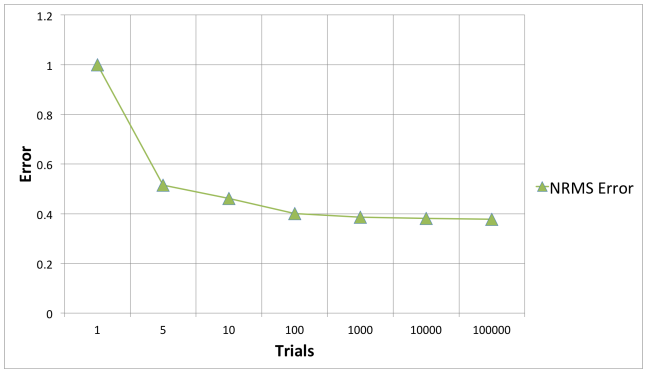

Figure 2: RMS Error vs Number of Trials

$n-1$. Each element has a timestamp associated with it to simulate user activity — for the purpose of this experiment, we assume that the timestamp is the array index. A lower array index indicates that the item is newer. Thus, we want to prefer items with a lower index in the output as these items indicate things that are done recently.

Using the differential privacy EM algorithm [36], we choose the scoring function to be maximized by returning a value with as low an array index as possible. Thus, we choose elements from the array with probability based on their array index.

In the experiments, we randomly generate the array and compute the score using the CD and the EM algorithms. We then plot the RMS and normalized RMS errors between these two algorithms. The error is the difference in the score returned by the CD and the EM algorithm. The CD algorithm will give us the "true" score; the EM algorithm (as it tries to preserve privacy) will give us a close approximation. We discuss the results in the following subsections.

5.3 RQ2 — Utility

For the first set of experiments, we varied the size of the array and plotted the RMS and normalized RMS errors between the CD and EM algorithms. The results are shown in Figure 1. To smooth out the noise in the experimental results (as CD is a deterministic algorithm while EM is a probabilistic one), we ran the experiment 1000 times with each array size and took averages. The graph shows us that as the size of the input array increases, the RMS error increases linearly — this is expected as with larger array sizes, the entries in the array have correspondingly larger values (due to our methodology), resulting in linearly increasing RMS error. Meanwhile, the normalized RMS error is roughly constant.

This shows us the tradeoff between accuracy and privacy. We observe that in these experiments, the loss of accuracy is relatively small — the normalized RMS error is less than 0.4. Thus, irrespective of the data set size, switching to the EM Algorithm (as required by our approach) from the CD Algorithm will not worsen the accuracy of the algorithm by more than the constant factor, and we have the added benefit that the EM algorithm also guarantees differential privacy. Whether the loss of accuracy is acceptable or not (or a worthy price to pay for the free privacy) is subjective and we deliberately do not enter a philosophical debate here (is accuracy of the system more "important" than user privacy? who decides this? the user? the web application developers?). Many papers in the database and theory communities have explored the tradeoffs between privacy and accuracy (*e.g.,* [9,19,35,36]) — our key point in this section is that yes, there is a loss of accuracy, but no worse than accepted in [35]. A limitation of our approach is that if this loss of accuracy is not acceptable for certain systems, our approach will not work.

For our second set of experiments, we varied the number of trials keeping the size of the array fixed to 1000. As the value computed using the EM algorithm is probabilistic in nature, we carry out multiple runs (called trials here) and take the average value over all the trials to smooth out the value. The graph plotting the RMS error vs the number of trials is shown in Figure 2. This graph shows us that as the number of trials increases, the RMS error reduces. Thus, initially, even though there may be a bigger error between the CD and EM algorithms, in the long run, the error will be small (but not zero, as a zero error would imply returning the accurate answer and thus, not preserving privacy).

With these set of experiments, we explored the utility of our approach. For an existing system (that may already use an algorithm similar to the CD one), a one-time change would be required to add in the EM algorithm and retrofit the system to our approach. This change is relatively straightforward and could even be automated. Making such a change, albeit results in a small loss of accuracy, gives the huge benefit of getting privacy for free without spending any additional computational resources.

5.4 RQ3 — Sustainability

For RQ3, we want to show the sustainability of our approach. With the EM algorithm in place, what we ideally want is that our system does not take any additional computational resources. We decided to use the CPU processing time to estimate the computational resources needed by the two algorithms. We instrumented the CD and EM algorithms and measured how long they took in the first set of experiments in Section 5.3 above. The resultant graph is shown in Figure 3. The graph shows us that for all data sizes the EM algorithm took less CPU time than the CD algorithm.

Not only does the EM algorithm not require any additional computational resources, it actually reduces the existing computation. Thus, changing to our approach will make the software system even more sustainable.

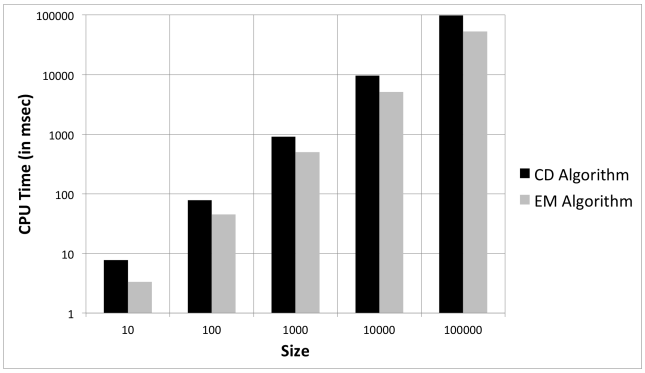

Figure 3: CPU Time (in msec) vs Size of data set

5.5 Threats to Validity

The notion of Differential Privacy may not relate to the user-centric view of Privacy as users might think it "strange" that the system assumes that bad things can happen anyway — the guarantee it gives is just regarding whether the user data is part of the system or not. While that is true, we feel that differential privacy has many compelling arguments in its favor — the biggest, for us, is not having to decide what data is sensitive and what is not. The differential privacy algorithms treat all data as sensitive making it easier not to leak data by accident. One would, therefore, not have to deal with the subjective nature of deciding what's sensitive. We also feel that the guarantee might actually make it even more compelling for the user. From their point of view — "if bad things are going to happen anyway, it's not going to hurt me much more if I participate.. so there's no harm in participating."

We used synthetic data in our evaluations rather than real-world data. For the research questions that we had — feasibility, utility, sustainability — synthetic data was sufficient. For Feasibility (RQ1), we use the theoretical proof from [36] so don't need data. For Utility (RQ2) and Sustainability (RQ3), we care only about the comparisons between the CD and the EM (and not the actual numbers in the experiments), so synthetic data — which was easier to work with — suffices. We would, however, need real data if we were doing, *e.g.,* surveys and our research question was if people thought the new system gave similar usability.

Finally, this work doesn't help in scenarios of non-temporal data access. We used the IMDB/Netflix examples earlier in the paper to make the general problem familiar to the reader; we address a special case of the problem where timestamps are available. In the differential privacy area, it's proven that for *any* method that has any utility, there exists side information that will break privacy on individual records. With differential privacy approaches such as the EM algorithm, the guarantees that exist for each individual are that participating in the database will not add to the risks that are already there.

6. RELATED WORK

Privacy has become an increasingly important topic for the community at large. A lot of different research communities are looking at the impact of privacy and techniques for improving privacy for users. Some examples of these communities are sociologists, computer scientists, HCI, etc. We discuss some of the relevant related work next.

Fang and LeFevre [25] proposed an automated technique for configuring a user's privacy settings in online social networking sites. Paul *et al.* [41] present using a color coding scheme for making privacy settings more usable. Squicciarini, Shehab, and Paci [46] propose a game-theoretic approach for collaborative sharing and control of images in a social network. Toubiana *et al.* [49] present a system that automatically applies users' privacy settings for photo tagging. All these papers propose new techniques that are targeted to making privacy settings "better" (*i.e.,* more usable, more visible) from a user's perspective. Our approach, on the other hand, targets the internal algorithms such as recommendations used by these systems.

There have been some recent papers on data privacy and software testing. Clause and Orso [15] propose techniques for the automated anonymization of field data for software testing. They extend the work done by Castro *et al.* [14] using novel concepts of path condition relaxation and breakable input conditions resulting in improving the effectiveness of input anonymization. Our work is orthogonal to the papers on input anonymization. The problem they address is — how can users anonymize sensitive information before sending it to the teams or companies that build the software? The problem we address is — how can systems that already have access to user data (such as purchase history, movie preferences, and so on) be engineered so that they don't leak sensitive information while doing data mining on the data? Further, the aim of our approach is to provide privacy "for free," *i.e.,* without spending extra computational resources on privacy. The input anonymization approaches require spending extra computation (between 2.5 minutes to 9 minutes) as they address a different problem. We believe that the our approach can be combined with the input anonymization approach if needed. If users are worried about developers at the company finding out sensitive information, input anonymization is essential. If, however, they are worried about accidental data leakage through the data mining of their information, using the "Privacy for Free" approach may be more suitable. This would also make the software system more sustainable as we don't spend any computation doing the anonymization of the inputs.

Taneja *et al.* [48] and Grechanik *et al.* [27] propose using k-anonymity [47] for privacy by selectively anonymizing certain attributes of a database for software testing. Their papers propose novel approaches using static analysis for selecting which attributes to anonymize so that test coverage remains high. Similar to above, our approach is orthogonal as we focus on an approach that will prevent accidental leakage of sensitive information via data mining or similar techniques. Further, these approaches using k-anonymity also require significant additional computational resources and thus, may not be sustainable when energy resources are scarce.

The testing problem above is concerned with internal data that users keep on their own computers and do not want to disclose outside their own computer (or put into a server and the testing is on that server software, but the data was understood to be specific to that user and never aggregated with other users). The problem we deal with instead is users who have intentionally disclosed data on a public system, entering their data via web browsers onto some website server

that is known to make publicly available certain data-mined community knowledge gleaned from aggregating that data with other users — but the users don't want their data to be personally identifiable from the aggregate.

Finally, work on input anonymization and k-anonymization both focus on software testing whereas our approach focuses on an approach for building privacy preserving systems or re-engineering existing software systems with minimal code changes (since only the parts affected need to be changed) with a specific goal — to make privacy sustainable and not require additional resources.

There has also been a lot of work related to data anonymization and building accurate data models for statistical use (*e.g.,* [2, 24, 34, 42, 51]). These techniques aim to preserve certain properties of the data (*e.g.,* statistical properties like average) so they can be useful in data mining while trying to preserve privacy of individual records. Similar to these, there are has also been work on anonymizing social networks [8] and anonymizing user profiles for personalized web search [57] The broad approaches include aggregating data to a higher level of granularity or adding noise and random perturbations. As we are interested in sustainable ways of achieving privacy, these approaches are not applicable as they typically require (a lot of) extra computational effort.

While there has been a lot of interest (and research) in data anonymization, we would like to reiterate that only data anonymization might not be enough. Narayanan and Shmatikov [40] demonstrate a relatively straightforward way of breaking the anonymity of data. They show how it is possible to correlate public IMDb data with private anonymized Netflix movie rating data resulting in the potential identification of the anonymized individuals. Backstrom *et al.* [3] also describe a series of attacks for de-anonymizing social networks that have been anonymized to be made available to the public. They describe two categories of attacks — active attacks where an evil adversary targets an arbitrary set of users and passive attacks where existing users try to discover their location in the network and thereby cause de-anonymization. Their results show that, with high probability and modest computational requirements, de-anonymization is possible for a real world social network (in their case, LiveJournal [12]). Finally, Zheleva and Getoor [56] show it's possible to infer private profiles of users on social networks based on their groups and friends.

7. DISCUSSION

The crux of this paper, and the novel idea, is that it is possible to combine two existing approaches to *increase* the degree of privacy in social computing systems, under certain conditions. This poses an interesting open problem — Are there other algorithms that we currently use for solving some problem that also accidentally provide privacy or some other added benefit?

A lot of research in the Theory and Cryptography community on Differential Privacy has focused on Mechanism Design [10, 21, 22, 43]. Mechanism Design is the process of coming up with new mechanisms that are differentially private and solve certain problems in domains such as machine learning and statistics. The previous sections hint at an interesting avenue of future research — Mechanism Discovery. We discovered how the CD algorithm as a side effect may provide differential privacy for free. It might be fruitful to look at currently used algorithms in varying domains and see if they too, as a side effect, provide differential privacy. This might lead to the discovery of generalized mechanisms for differential privacy that can be used in other domains, which have not yet been proposed or discovered by Theory and Cryptography researchers. Mechanism Discovery might act as a great complement to the Mechanism Design research.

In order for Mechanism Discovery to be successful, a greater emphasis must be placed on Multidisciplinary research. Even though there is some research in recommender system privacy [7, 45], most of the papers do not use a formal and precise definition of privacy. Our community could benefit a lot from the precise and formal use of differential privacy. Similarly, most of the Theory and Cryptography community may not be aware of the privacy research done by our, or other, communities. There might be a lot of interesting discoveries of mechanisms suitable for differential privacy. The only way any of this can be achieved is by a greater emphasis on Multidisciplinary research using areas such as systems, theory, cryptography, web, and databases.

8. CONCLUSION

As social computing systems that collect users' data proliferate, privacy has and will continue to become a major concern for the society at large. The main research question that we wanted to answer is — Is there an approach that can be used with a certain web applications and software systems, that will achieve privacy without spending any extra resources on computational overhead? Our "Privacy for Free" approach can achieve privacy as an accidental and beneficial side effect of addressing concept drift. The results of our evaluations show the feasibility, utility, and in particular, the sustainability of our approach as it does not require any additional computational resources to guarantee privacy.

9. ACKNOWLEDGMENTS

Sheth and Kaiser are members of the Programming Systems Laboratory is funded in part by NSF CCF-1161079, NSF CNS-0905246, and NIH U54 CA121852. Malkin is a member of the Crypto Lab, funded in part by NSF 0831094 and 0347839 and DHS N66001-09-C-0080.

10. REFERENCES

[1] N. R. Adam and J. C. Worthmann. Security-control methods for statistical databases: a comparative study. *ACM Comput. Surv.*, 21(4):515–556, 1989.

[2] D. Agrawal and C. C. Aggarwal. On the design and quantification of privacy preserving data mining algorithms. In *PODS '01: Proceedings of the twentieth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 247–255, New York, NY, USA, 2001. ACM.

[3] L. Backstrom, C. Dwork, and J. Kleinberg. Wherefore art thou r3579x?: anonymized social networks, hidden patterns, and structural steganography. In *WWW '07: Proceedings of the 16th international conference on World Wide Web*, pages 181–190, New York, NY, USA, 2007. ACM.

[4] M. Barbaro, T. Zeller, and S. Hansell. A face is exposed for AOL searcher no. 4417749. *New York Times*, 9, 2006.

[5] L. L. Beck. A security mechanism for statistical database. *ACM Trans. Database Syst.*, 5(3):316–3338, 1980.

[6] A. Begel, K. Y. Phang, and T. Zimmermann. Codebook: discovering and exploiting relationships in software repositories. In *ICSE '10: Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering*, pages 125–134, New York, NY, USA, 2010. ACM.

[7] S. Berkovsky, Y. Eytani, T. Kuflik, and F. Ricci. Enhancing privacy and preserving accuracy of a distributed collaborative filtering. In *RecSys '07: Proc. of the 2007 ACM conf. on Recommender systems*, pages 9–16, 2007.

[8] S. Bhagat, G. Cormode, B. Krishnamurthy, and D. Srivastava. Privacy in dynamic social networks. In *Proceedings of the 19th international conference on World wide web*, WWW '10, pages 1059–1060, New York, NY, USA, 2010. ACM.

[9] A. Blum, C. Dwork, F. McSherry, and K. Nissim. Practical privacy: the sulq framework. In *PODS '05: Proceedings of the twenty-fourth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 128–138, New York, NY, USA, 2005. ACM.

[10] A. Blum, K. Ligett, and A. Roth. A learning theory approach to non-interactive database privacy. In *STOC '08: Proceedings of the 40th annual ACM symposium on Theory of computing*, pages 609–618, New York, NY, USA, 2008. ACM.

[11] B. Bosker. Facebook CEO 'Doesn't Believe In Privacy'. `http://www.huffingtonpost.com/2010/04/29/zuckerberg-privacy-stance_n_556679.html`, April 2010.

[12] Brad Fitzpatrick. LiveJournal. `http://www.livejournal.com/`, 1999.

[13] G. Canfora, L. Cerulo, M. Cimitile, and M. Di Penta. Social interactions around cross-system bug fixings: the case of freebsd and openbsd. In *Proceeding of the 8th working conference on Mining software repositories*, MSR '11, pages 143–152, New York, NY, USA, 2011. ACM.

[14] M. Castro, M. Costa, and J.-P. Martin. Better bug reporting with better privacy. In *Proceedings of the 13th international conference on Architectural support for programming languages and operating systems*, ASPLOS XIII, pages 319–328, New York, NY, USA, 2008. ACM.

[15] J. Clause and A. Orso. Camouflage: automated anonymization of field data. In *Proceeding of the 33rd international conference on Software engineering*, ICSE '11, pages 21–30, New York, NY, USA, 2011. ACM.

[16] E. Cohen and M. Strauss. Maintaining time-decaying stream aggregates. In *Proc. of the 22nd ACM SIGMOD-SIGACT-SIGART symposium on principles of database systems (PODS)*, pages 223–233, 2003.

[17] T. Dalenius. Towards a methodology for statistical disclosure control. *Statistik Tidskrift*, 15:429–444, 1977.

[18] Y. Ding and X. Li. Time weight collaborative filtering. In *CIKM '05: Proceedings of the 14th ACM international conference on Information and knowledge management*, pages 485–492, New York, NY, USA, 2005. ACM.

[19] I. Dinur and K. Nissim. Revealing information while preserving privacy. In *PODS '03: Proceedings of the twenty-second ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 202–210, New York, NY, USA, 2003. ACM.

[20] C. Dwork. Differential privacy. *IN ICALP*, 2:1–12, 2006.

[21] C. Dwork, F. McSherry, K. Nissim, and A. Smith. Calibrating noise to sensitivity in private data analysis. *Theory of Cryptography*, pages 265–284, 2006.

[22] C. Dwork, M. Naor, O. Reingold, G. N. Rothblum, and S. Vadhan. On the complexity of differentially private data release: efficient algorithms and hardness results. In *STOC '09: Proceedings of the 41st annual ACM symposium on Theory of computing*, pages 381–390, New York, NY, USA, 2009. ACM.

[23] C. Dwork and K. Nissim. Privacy-preserving datamining on vertically partitioned databases. *Lecture Notes in Computer Science*, pages 528–544, 2004.

[24] A. Evfimievski, J. Gehrke, and R. Srikant. Limiting privacy breaches in privacy preserving data mining. In *PODS '03: Proceedings of the twenty-second ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 211–222, New York, NY, USA, 2003. ACM.

[25] L. Fang and K. LeFevre. Privacy wizards for social networking sites. In *Proceedings of the 19th international conference on World wide web*, WWW '10, pages 351–360, New York, NY, USA, 2010. ACM.

[26] D. Fletcher. How Facebook Is Redefining Privacy. `http://www.time.com/time/business/article/0,8599,1990582.html`, May 2010.

[27] M. Grechanik, C. Csallner, C. Fu, and Q. Xie. Is data privacy always good for software testing? *Software Reliability Engineering, International Symposium on*, 0:368–377, 2010.

[28] J. L. Herlocker, J. A. Konstan, L. G. Terveen, and J. T. Riedl. Evaluating collaborative filtering recommender systems. *ACM Trans. Inf. Syst.*, 22:5–53, January 2004.

[29] F. Heylighen and J. Bollen. Hebbian algorithms for a digital library recommendation system. *Parallel Processing Workshops, International Conference on*, 0:439, 2002.

[30] S. Johnson. Web Privacy: In Praise of Oversharing. `http://www.time.com/time/business/article/0,8599,1990586.html`, May 2010.

[31] R. Klinkenberg. Learning drifting concepts: Example selection vs. example weighting. *Intell. Data Anal.*, 8(3):281–300, 2004.

[32] Y. Koren. Collaborative filtering with temporal dynamics. *Commun. ACM*, 53(4):89–97, 2010.

[33] I. Koychev and I. Schwab. Adaptation to drifting user's interests. In *Proceedings of ECML2000 Workshop: Machine Learning in New Information Age*, pages 39–46. Citeseer, 2000.

[34] N. Lathia, S. Hailes, and L. Capra. Private distributed

collaborative filtering using estimated concordance measures. In *RecSys '07: Proceedings of the 2007 ACM conference on Recommender systems*, pages 1–8, New York, NY, USA, 2007. ACM.

[35] F. McSherry and I. Mironov. Differentially private recommender systems: building privacy into the net. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '09, pages 627–636, New York, NY, USA, 2009. ACM.

[36] F. McSherry and K. Talwar. Mechanism design via differential privacy. In *FOCS '07: Proceedings of the 48th Annual IEEE Symposium on Foundations of Computer Science*, pages 94–103, Washington, DC, USA, 2007. IEEE Computer Society.

[37] Microsoft. *Green Computing*, volume 18. The Architecture Journal, 2008.

[38] C. Murphy, S. Sheth, G. Kaiser, and L. Wilcox. genSpace: Exploring Social Networking Metaphors for Knowledge Sharing and Scientific Collaborative Work. In *1st Intl. Workshop on Social Software Engg. and Applications*, pages 29–36, September 2008.

[39] S. Murugesan. Harnessing green it: Principles and practices. *IT Professional*, 10(1):24 –33, jan.-feb. 2008.

[40] A. Narayanan and V. Shmatikov. How to break anonymity of the netflix prize dataset. *CoRR*, abs/cs/0610105, 2006.

[41] T. Paul, M. Stopczynski, D. Puscher, M. Volkamer, and T. Strufe. C4ps: colors for privacy settings. In *Proceedings of the 21st international conference companion on World Wide Web*, WWW '12 Companion, pages 585–586, New York, NY, USA, 2012. ACM.

[42] H. Polat and W. Du. Privacy-preserving collaborative filtering using randomized perturbation techniques. In *Data Mining, 2003. ICDM 2003. Third IEEE International Conference on*, pages 625–628, Nov. 2003.

[43] A. Roth and T. Roughgarden. Interactive privacy via the median mechanism. In *STOC '10: Proceedings of the 42nd ACM symposium on Theory of computing*, pages 765–774, New York, NY, USA, 2010. ACM.

[44] M. Shiels. Germany officials launch legal action against Facebook. `http://news.bbc.co.uk/2/hi/technology/8798906.stm`, July 2010.

[45] R. Shokri, P. Pedarsani, G. Theodorakopoulos, and J.-P. Hubaux. Preserving privacy in collaborative filtering through distributed aggregation of offline profiles. In *RecSys '09: Proceedings of the third ACM conference on Recommender systems*, pages 157–164, New York, NY, USA, 2009. ACM.

[46] A. C. Squicciarini, M. Shehab, and F. Paci. Collective privacy management in social networks. In *Proceedings of the 18th international conference on World wide web*, WWW '09, pages 521–530, New York, NY, USA, 2009. ACM.

[47] L. Sweeney. k-anonymity: a model for protecting privacy. *Int. J. Uncertain. Fuzziness Knowl.-Based Syst.*, 10(5):557–570, 2002.

[48] K. Taneja, M. Grechanik, R. Ghani, and T. Xie. Testing software in age of data privacy: a balancing act. In *Proceedings of the 19th ACM SIGSOFT symposium and the 13th European conference on Foundations of software engineering*, SIGSOFT/FSE '11, pages 201–211, New York, NY, USA, 2011. ACM.

[49] V. Toubiana, V. Verdot, B. Christophe, and M. Boussard. Photo-tape: user privacy preferences in photo tagging. In *Proceedings of the 21st international conference companion on World Wide Web*, WWW '12 Companion, pages 617–618, New York, NY, USA, 2012. ACM.

[50] U.S. Energy Information Administration. International Energy Outlook 2010 - Highlights. `http://www.eia.doe.gov/oiaf/ieo/highlights.html`, May 2010.

[51] V. S. Verykios, E. Bertino, I. N. Fovino, L. P. Provenza, Y. Saygin, and Y. Theodoridis. State-of-the-art in privacy preserving data mining. *SIGMOD Rec.*, 33(1):50–57, 2004.

[52] J. Vidal. The end of oil is closer than you think. `http://www.guardian.co.uk/science/2005/apr/21/oilandpetrol.news`, April 2005.

[53] J. Whitehead. Collaboration in software engineering: A roadmap. In *2007 Future of Software Engineering*, FOSE '07, pages 214–225, Washington, DC, USA, 2007. IEEE Computer Society.

[54] G. Widmer and M. Kubat. Learning in the presence of concept drift and hidden contexts. *Machine Learning*, 23(1):69–101, 1996.

[55] J. Zhang and P. Pu. A recursive prediction algorithm for collaborative filtering recommender systems. In *RecSys '07: Proc. of the 2007 ACM conference on Recommender systems*, pages 57–64, 2007.

[56] E. Zheleva and L. Getoor. To join or not to join: the illusion of privacy in social networks with mixed public and private user profiles. In *Proceedings of the 18th international conference on World wide web*, WWW '09, pages 531–540, New York, NY, USA, 2009. ACM.

[57] Y. Zhu, L. Xiong, and C. Verdery. Anonymizing user profiles for personalized web search. In *Proceedings of the 19th international conference on World wide web*, WWW '10, pages 1225–1226, New York, NY, USA, 2010. ACM.

[58] M. Zuckerberg. Making Control Simple. `http://blog.facebook.com/blog.php?post=391922327130`, May 2010.