Crowd Mining

Yael Amsterdamer
Tel Aviv University
Tel Aviv, Israel
yaelamst@post.tau.ac.il

Yael Grossman
Tel Aviv University
Tel Aviv, Israel
yaelgros@post.tau.ac.il

Tova Milo
Tel Aviv University
Tel Aviv, Israel
milo@cs.tau.ac.il

Pierre Senellart
Télécom ParisTech & The University of Hong Kong
Paris, France & Hong Kong
pierre.senellart@telecom-paristech.fr

ABSTRACT

Harnessing a crowd of Web users for data collection has recently become a wide-spread phenomenon. A key challenge is that the human knowledge forms an open world and it is thus difficult to know what kind of information we should be looking for. Classic databases have addressed this problem by data mining techniques that identify interesting data patterns. These techniques, however, are not suitable for the crowd. This is mainly due to properties of the human memory, such as the tendency to remember simple trends and summaries rather than exact details.

Following these observations, we develop here for the first time the foundations of crowd mining. We first define the formal settings. Based on these, we design a framework of generic components, used for choosing the best questions to ask the crowd and mining significant patterns from the answers. We suggest general implementations for these components, and test the resulting algorithm's performance on benchmarks that we designed for this purpose. Our algorithm consistently outperforms alternative baseline algorithms.

Categories and Subject Descriptors

H.2.8 [**Database Applications**]: Data mining

Keywords

crowd mining, crowdsourcing, association rule learning

1. INTRODUCTION

Problem. Social scientists, micro-economists, journalists, marketers, public health specialists, and politicians alike routinely analyze people's behaviors to find new trends, understand and document behaviors, and possibly put forward adequate policies in response. Discovering statistically significant patterns in the crowd's habits is however a challenging task, and traditional tools (interviews, polls, surveys) to collect data from individuals about their daily life are costly to implement, and moreover, it is often hard to know which are the best questions to ask. This is in part due to the fact that human knowledge forms an *open world* [13]; it is often the case that one wants to use the crowd to find out what is *interesting and significant* about a particular topic, without full knowledge about what the topic consists of.

In traditional settings where one has access to a database of all relevant information, the problem of finding significant, as-yet-unknown patterns has been addressed via *data mining*. In particular, *association rules* [1], which capture when one set of data items indicates the presence (or values) of another set, were shown to be useful in identifying significant patterns. The typical application is shopping baskets: a store can, by analyzing purchase records, mine rules about which items are bought together. In this traditional setting, the significant rules are assumed to be initially unknown, and data mining algorithms dynamically construct database queries to identify them, relying on indicators such as *support* and *confidence*.

However, association rule mining is based on the assumption that *transactions* (e.g., sets of items bought together in the shopping basket application) are available. When dealing with the crowd's behavior, it is often impossible to have access to these transactions, i.e., to ask individuals for extensive descriptions of their past activities.

As a motivating example, consider a health researcher, Alice, who is interested in discovering and testing new drugs by analyzing the practices of folk medicine (also known as traditional medicine, i.e., medicinal practice that is neither documented in writing nor tested out under a scientific protocol). Alice is interested in discovering association rules such as "Garlic and oregano leaves can be used to treat a sore throat" or "Chamomile tea drinkers tend to experience sleepiness" together with their support and confidence. Alice can interview folk healers, but it is unrealistic for her to ask them for an exhaustive list of all cases they treated over the years. A database of transactions, each corresponding to the symptoms and treatments of a particular illness case, simply cannot be constructed. As another example, consider a social scientist analyzing life habits of people, in terms of activities (watching TV, jogging, reading, etc.) correlated with their contexts (time of the day, weather, presence of friends, etc.). Here too it is impossible to obtain a comprehensive database of transactions, representing all the "events" (mornings, evenings…) with their context and all the activities performed in them, for a large community. Or consider a public health researcher studying a population's eating habits to understand how best to fight an obesity epidemics (in which case transactions correspond to types of food eaten or cooked together); an ergonomics consultant who surveys work habits in a company to detect positive or negative patterns in work organization that should be encouraged or corrected (transactions correspond to events in the day-to-day work life, e.g., a particular meeting with the communication means used for it). In all these cases, we deal with an open world (no pre-existing list of relevant activities, foodstuff, or work environment factors) and it is impossible to obtain an exhaustive database of all transactions.

Crowd-based data sourcing is a new and powerful data procurement paradigm that engages Web users to collectively contribute data [12]. Information is often gathered by posing *questions* to users, which they may answer for some small payment, for social or moral reasons, or in the context of a game [17, 25]. Even though people cannot recall all of their transactions, social studies show that they can often provide simple summaries, and even more complicated ones when asked targeted questions [5]. Summaries can be viewed as *rules that apply to individuals*, or personal rules. As an example, a folk healer may be able to tell Alice, when asked which symptoms he usually encounters (an *open question*), that he deals with nausea cases almost every week. He can also answer perhaps more complex but targeted questions (named *closed questions*), such as "When a patient has both headaches and fever, how often do you use a willow tree bark infusion to treat him?" The answer might be something like "A few times a year, that is, perhaps for a half of all such patients." Mining the crowd thus differs from traditional data mining in two fundamental ways: on the one hand, one cannot have access to the transactions in extension, but on the other hand, summaries allow obtaining, with one request, information that encapsulates many transactions.

The problem we address is thus: *How to exploit the unique capabilities of the crowd to mine data from the crowd?*

Note that even when transactions are partly available, we can use the crowd to gather complementary information: a store might have easy access to its consumers' shopping baskets and use classic mining approaches on them, but it does not know what its consumers buy outside of its stores and cannot grasp the subjective consumer experience; for this, stores and marketing agencies use consumer panels and surveys. However, these may be time-consuming, expensive and inaccurate, and we would like to use crowdsourcing for a more efficient alternative.

If people are aware of the rules that apply to them, why mine these rules? First and most importantly, our goal is to identify trends in the entire population. Each person is only familiar with her own habits, which form partial information about the general trends. Mining many users allows us to piece together the complete picture. Second, it is hard for people to list even their personal rules – only prominent rules may be recalled in this manner [5]. Our mining process asks users concrete, well-defined questions, which help digging deeper into their memory.

Contributions. To tackle the crowd mining problem, we present in this article the following contributions:

1. We define a model for the average behavior of the crowd. We model the types of questions a user may be asked (open and closed), and the data we retrieve as a result. This relies on per-user notions if pattern significance. The formal model allows defining the ultimate goal of crowd mining: finding the association rules that are *overall* significant. This highlights the required changes in the data mining paradigm to make it suitable for the crowd. (Section 2)
2. We present an effective crowd-mining framework. The generic components of this framework can be used for iteratively choosing the best crowd questions, processing the answers, and upon demand, deciding which rules are the significant ones. Some framework components are "black boxes" which allow plugging in different implementations for different applications. (Section 3)
3. Since computing the exact overall significance of a rule would require posing questions to all the users, it is infeasible for large crowds. As an important component of our framework, we perform a probabilistic modeling of the crowd and their behavior, and develop a formulation for well-founded probabilistic estimates of the significance of a rule, the potential error in this estimation, and the impact of asking a given question on this significance and uncertainty. (Section 4)
4. We also propose a specific implementation of the remaining components of the framework, based on adaptations of classic mining algorithms [2], and general techniques from sequential sampling [24]. The unique settings of crowd mining prevent us from using existing data mining algorithms as-is. The main reason is that at each point we may have only discovered a subset of the items domain and, for the rules composed of the items discovered so far, estimations are always inaccurate to some degree. Our solution also serves as an illustration of how to adapt classic association rule mining algorithms to the crowd. (Section 5)
5. Finally, we propose benchmark datasets that enable systematic evaluation of crowd-mining algorithms performance. We show, by means of an extensive experimental evaluation on an implementation of our algorithm, that it consistently outperforms alternative baseline algorithms, and identifies significant rules with greater accuracy, while asking fewer questions. The results also demonstrate the viability of our crowd-mining framework, even with simple implementations of some of the black boxes. (Section 6)

We discuss related work in Section 7 and conclude with possible extensions in Section 8.

2. PRELIMINARIES

We are interested in asking users about personal rules, and inferring about overall important rules and general trends. For that, we start with basic definitions of association rules and their quality measurements per individual. We claim that these simple definitions, based on classic association rule theory [1], can capture the summarized manner in which people tend to record certain types of information. This also guides us in the types of questions we ask the crowd and the interpretation we give to their answers.

Let $\mathcal{U}$ be a set of human users, and let $\mathcal{I} = \{i_1, i_2, i_3, \dots\}$ be a non-empty finite set of unique items. Define a *transaction* t as a subset of $\mathcal{I}$. Each user u is associated with a *personal database* D_u, which is a bag (multiset) of transactions. $|D_u|$ denotes the number of transactions in D_u.

Let $A, B \subseteq \mathcal{I}$, $A \cap B = \emptyset$. Then $A \rightarrow B$ is an *association rule*, which may be used to signify that in a personal database D_u, $A \subseteq t$ implies $B \subseteq t$, for any $t \in D_u$. From this point we sometimes use the notation $a, b \to c, d$ instead of $\{a, b\} \to \{c, d\}$, for brevity.

When mining folk medicine, for instance, items may represent symptoms, treatments and medicine ingredients. An association rule might be flu$\to$garlic, signifying that when a user has the flu, she takes garlic.

In data mining, rules are considered interesting when they occur frequently in the data. We recall the standard definitions of support and confidence, which are used here as indicators for rule significance *per user*.

Definition 2.1 (User Confidence and Support). *Given a user u with database D_u and a rule $A \to B$, let the* user support *of $A \to B$ in the transactions of u be defined as:*

$$supp_u(A \to B) := \frac{|\{t \in D_u \mid A, B \subseteq items(t)\}|}{|D_u|}$$

and similarly, define the user confidence *of $A \to B$ as:*

$$conf_u(A \to B) := \frac{|\{t \in D_u \mid A, B \subseteq items(t)\}|}{|\{t \in D_u \mid A \subseteq items(t)\}|}$$

Questions and answers. Data mining techniques generally rely on processing transactions for association rule learning. While we model the knowledge of every user from the crowd as an underlying personal database of transactions, this database is *completely virtual*, and not available to us for mining directly. For instance, a transaction in folk medicine may represent a particular case of the flu with all the symptoms and treatments used. People usually do not remember many such elaborate details, and instead tend to remember summaries, which can be viewed as rules.

Consequently, we give a definition for the method of accessing personal databases based on association rules (rather than transactions) and their user support and confidence. For a given rule, users can tell us the significance of the rule in their personal databases. Moreover, users can spontaneously recall rules. Thus, the two following types of questions to users are considered.

- **Closed questions.** Questions modeled as $A \overset{?}{\to} B$. We interpret the answer of a user u as the support and confidence of the rule $A \to B$ w.r.t. D_u, i.e., the pair $\langle supp_u(A \to B), conf_{D_u}(A \to B)\rangle$.
- **Open questions.** Questions modeled as $? \overset{?}{\to} ?$. The answer of a user u is interpreted as some A, B, along with the confidence and support of $A \to B$ in D_u.

We show in the sequel how these types of questions can be used for mining interesting rules from the crowd.

In order to be presented to users, questions must be phrased in natural language. For instance, consider the closed question flu$\overset{?}{\to}$garlic. One way of displaying it is by two natural language questions, corresponding to the support of "flu" and the confidence of flu$\overset{?}{\to}$garlic, respectively: "How often do you usually have the flu?" and "When you have the flu, how often do you take garlic?" An open question may be formed as "Tell us about an illness, the way you treat it and how often this occurs." Users may then enter their answer in natural language, e.g., "When I have the flu I take garlic about 80% of the time. This usually happens twice a year." In such a case, NLP tools must be used for interpreting the answer. Alternatively, users may also be given multiple options to choose from (e.g., "always", "often", "sometimes", "rarely"); or fill in the frequency by providing a number of occurrences and the time period from closed lists.

The way in which people answer the questions in practice reveals another challenge: people mostly give only *rough estimations* for the support and confidence of rules. For instance, when the exact support is 0.879 they may give an approximate answer ("about 90% of the time"), a range ("80–100% of the time") or even be slightly off the correct answer ("90–100% of the time") [5]. In our definitions we assume that both user support and confidence are provided to us as a single, perhaps approximate, number.[1] One way of modeling the error in user answers might be, e.g., assuming some error range per user. As we explain next, our choice of overall quality measures for rules allows us to ignore individual user errors and still obtain good estimations.

Note that support and confidence of *different* association rules may be dependent. For instance, the support of $A \cup B \to C$ is bounded by the support of $A \to C$. We consider these dependencies in asking about rules for which we have no samples (see Section 5), but not in our approximations (for computation simplicity). This means that we gather information on each rule separately from the crowd. Note that this does not affect the algorithm correctness, but gathering information on several rules at once could be a further optimization (see extensions in Section 8).

Overall Rule Significance. We are interested in inferring general trends, i.e., rules that have overall significance in a group of users. For that, we first need to choose how to *aggregate* user answers into an overall quality measure per rule. Since our focus in this work is on users rather than transactions, we define significant rules as rules with high *average* user support and confidence.[2] Formally:

Definition 2.2 (Significant Rules). *Let $\mathcal{U}$ be a group of users. We say that a rule $r = A \to B$ is* significant *in $\mathcal{U}$ if it satisfies $\mathrm{avg}_{u \in \mathcal{U}}\, supp_u(r) \geqslant \Theta_s$ and $\mathrm{avg}_{u \in \mathcal{U}}\, conf_u(r) \geqslant \Theta_c$, where $0 \leqslant \Theta_s, \Theta_c \leqslant 1$ are predetermined threshold values.*

Note that in order to obtain the true average support and confidence one must ask every user in $\mathcal{U}$ about r. In practice this is impossible, since the number of relevant users and rules may be huge. Thus, we resort to sample-based empirical estimations. As described in Section 4, the choice of average as the aggregation function allows us to employ rigorous statistical tools for such estimations.

Another advantage of choosing avg here is for handling the error in individual user answers. In many case studies, it was shown that on average, individual user errors "cancel" each other out such that the average of the answers is very close to the truth. This phenomenon forms the foundations for the theory of the *wisdom of the crowd* [22]. It is particularly relevant in situations when user answers are independent, and indeed, as the answers we seek are personal, there is a good reason to assume such independence.

The choice of thresholds for the average support and confidence in Definition 2.2 is taken from classic data mining. We note that the solution we present can be adapted to support other functions on the average support and confidence, e.g., a threshold on their sum or product.

[1] Answers that describe the frequency of habits may be interpreted w.r.t. a fixed time period. For instance, "x times a week" may be interpreted as $\frac{x}{7}$.

[2] Other possible aggregates are discussed in Section 8.

Choice of users. Different works in the area of crowdsourcing focus either on the choice of questions, the choice of users to ask, or both [19, 8, 4]. While the ability to choose the questions to ask the users always exists, the ability to choose users is inherently limited. This is due to the fact that particular users may not always be available, or willing to answer more than a few questions. We therefore make the *least restrictive assumption*, namely that the users who answer our questions are chosen randomly each time. As we show next, this assumption is actually useful in our settings, since it allows treating user answers as random samples.

3. FRAMEWORK

Having identified the general settings in which our crowd mining algorithm works, we turn to design the general structure of a novel crowd-mining framework. This framework is composed of theoretical components that are used as *black-boxes*. Depending on prior knowledge, particular estimations about the data distribution, a different choice of the target function, etc. – different *implementations* of the black-boxes may be preferred. This observation is not unique to crowd mining, and also applies to different data mining algorithms, machine learning algorithms, etc.

The framework structure we suggest is designed to facilitate a crowd-mining algorithm. A few unique properties of crowd mining (as opposed to classic data mining) guide the course of this algorithm, as follows.

1. **Using the crowd wisely.** The most costly resource in our settings, both in latency and possibly monetary cost, are the crowd questions. Thus, we aim to minimize the number of questions by maximizing the knowledge gain per question.
2. **Data access model.** As defined in Section 2, a user answer gives us information only w.r.t. a particular rule. Thus, the algorithm is directed by the choice of rules to ask about.
3. **Estimated quality.** As it is impossible to ask all users each question, our algorithm relies on estimations for rule significance, as well as for the effect of a question on our confidence.

Based on these characteristics, our algorithm is an *interactive* one, and works in iterations. At each iteration, we use the knowledge gained so far to decide which question to ask next. The question should be chosen so as to maximize knowledge gain. This is the most challenging part of the algorithm, and the main role of the component framework. Once the user answer is obtained, the framework components can be used to update our estimations for the significance of different rules. In addition, at each point of the algorithm, we can rely on these estimations for computing and returning the set of significant rules.

Remark. For simplicity, we assume that the algorithm only selects *one question at a time*. We briefly discuss the selection of the best bulk of K next questions, which may be then posed to the crowd in parallel, in Section 8.

Our framework components are organized in a *hierarchy*. Each black-box implementation in this hierarchy may use lower-tier black-boxes. If the implementation of this black-box is replaced, the lower-level black-boxes may not be required anymore. An overview of the hierarchy of our framework components is given in the diagram in Figure 1. The boxes with light background are boxes for which the

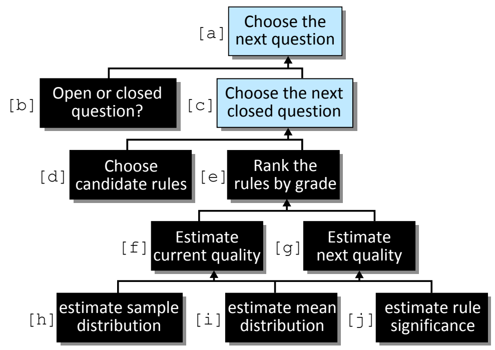

[a] Choose the next question
[b] Open or closed question?
[c] Choose the next closed question
[d] Choose candidate rules
[e] Rank the rules by grade
[f] Estimate current quality
[g] Estimate next quality
[h] estimate sample distribution
[i] estimate mean distribution
[j] estimate rule significance

Figure 1: Framework Component Hierarchy

```
Data: Set of users $\mathcal{U}$, a budget $N$
Result: $S$: a set of association rules
Initialize Enabled [d];
for every $i \in \{1, \ldots, N\}$ OR until stopped do
    $u \leftarrow$ a random user;
    if Ask open question? [b] then
        $q \leftarrow$ an open question;
        $(r', s, c) \leftarrow$ the answer of $u$ for $q$;
    else
        $r' \leftarrow \arg\max_{r \in Enabled}\, grade(r)$ [e];
        $q \leftarrow$ a closed question about $r'$;
        $(s, c) \leftarrow$ the answer of $u$ for $q$;
    Update Enabled [d];
    Add $(s, c)$ to the samples of $r'$;
$S \leftarrow \emptyset$ ;
for every $r \in Enabled$ do
    if $r$ is significant [j] then
        Add $r$ to $S$;
return $S$;
```
Algorithm 1: Crowd-mining association rules

implementation is straightforward, using the boxes of the level below. The black-box components are colored black.

Algorithm 1 gives a high-level description of a crowd-mining algorithm using the framework components. The use of a particular component is indicated by marking the relevant lines with the component letter. Some of the components are called implicitly, such as in the call to the *grade* function [e], which involves components [f-j], or the choice of the next question [a] which appears decomposed and spans over the **if-then-else** part inside the loop.

The course of the algorithm. Given a budget of N questions to pose to the crowd, the algorithm completes at most N iterations. It may be stopped at any point, e.g., if too much time has elapsed or if we have obtained satisfactory accuracy. At each iteration we first choose a random user u, according to our assumption from Section 2. Then, we use the framework components to decide which question to ask.

Note that, as in many other learning problems, we have a tradeoff between obtaining (possibly) new information by asking open questions, and refining the accuracy of our estimations by asking closed questions. In particular, items of the domain, which are initially unknown, can only be discovered by open questions, thus expanding the set of rules which we can compose and evaluate. On the other hand, the fact that we can cleverly select where to refine our knowledge (by asking closed questions) gives us the advantage over random sampling (asking only open questions). Thus, the first black-box component used in the algorithm is responsible for choosing between closed and open questions (box [b]), while aiming to balance the tradeoff. Its particular implementation may depend on assumptions about answers to open questions (according to which distribution they are selected), and on the need in exploration (e.g., how much of the item domain has been discovered).

If the chosen type is *open*, an open question is posed to the user u, who returns a rule r' with its support and confidence (according to Section 2).

The case of a *closed* question is the more algorithmically challenging one. When selecting the next closed question to ask, we do not want to consider all the possible rules that can be composed of the known data items. This is because 1) there is a huge amount of rules, and 2) most of them are not interesting. Instead, we can focus on a small subset of more interesting rules, called *enabled* rules (box [d]). The choice of which rules to enable can be inspired by the rule exploration order of existing data mining algorithms (See Section 5). However, it must also take into account unique features of the crowd mining scenario, such as the fact that no full information about a rule is ever collected.

Now, among the enabled rules, we ask about the rule with the highest *grade* (box [e]). The grade is computed in an elaborate way so as to reflect the effect of asking a question on r'. We would like to choose the rule that, if this were the final iteration, would maximize our *overall output quality*. After selecting a rule r', we ask the user u about it and record his answer (user support and confidence).

At the end of Algorithm 1, the set of significant rules is computed and returned as the output. Component [j] is reused here for finding significant rules.

The lower tiers. We now turn to explain the lower level components in the hierarchy, which are used in the implementation of box [e]. See Section 4, where we describe the non-trivial implementation for these components, which is a key contribution of this paper.

The second-lowest-level components in Figure 1 refer to estimation of the current quality of the algorithm output (box [f]), and the "next" quality, namely the quality after asking one more question on a particular rule r' (box [g]). The difference between those measures, i.e., the effect of the answer about r' on the quality, may be used for rule ranking.

To estimate the quality before and after asking about r', we must first estimate the overall support and confidence of each rule (recall Definition 2.2). For generality, instead of computing just one estimation, we compute a *distribution* that reflects the probability for different values of the means, given the collected knowledge (box [i]). Next, we decide whether r' is significant (box [j]). The decision method should minimize the probability of making an error. Finally, the estimation of the next quality depends on the next user answer about r' (box [h]). We can use the samples (user answers) and any standard fitting method for estimating the distribution of the next user answer.

4. ERROR ESTIMATION, RULE RANKING

In order to rank closed questions (box [e]), we need to predict the expected effect of each answer on the overall output quality. We suggest, as a possible quality measure, the *expected total number of errors* (to be formally defined in the sequel). We show that this simple option proves to be a useful and practical one.

Table 1: Running Example for rule $\hat{r}$

	current	**next (expected)**
Sample mean	$\mu = (0.5125, 0.5125)$	μ
Sample covariance	$\Sigma = 10^{-2} \cdot \begin{pmatrix} 1.72 & 0.39 \\ 0.39 & 1.72 \end{pmatrix}$	Σ
$f_{\hat{r}}$ covariance	$\frac{\Sigma}{4}$	$\frac{\Sigma}{5}$
$P_{\text{err}}(\hat{r})$	0.367	0.332

We next describe in detail a key contribution of this paper, namely, suggested implementations for components [e-j], which work together to minimize the expected output error. For generality, our choices are based on the assumption of *zero prior knowledge*, and some of them include parameters that may be tweaked experimentally. This way, and as we show in our experiments (See Section 6), we construct a general-purpose crowd-mining algorithm. The implementations are explained in a bottom-up fashion, with the letter of the relevant component next to each section title.

Estimating the mean [i]. According to Section 2, we are interested in approximating the average support and confidence. When we ask a closed question about a particular rule r, we in fact choose a user randomly (uniformly) out of a huge number of users, and obtain his user support and confidence for r. This may be modeled as *sampling from an unknown distribution* $\tilde{g}_r$, the distribution of the user answers. Each sample is a 2D vector, where the first dimension is the support and the second is the confidence. According to the central limit theorem, the sample mean, denoted f_r, approaches a normal distribution. We thus approximate the distribution of the sample mean f_r by the bivariate normal distribution with mean vector $\tilde{\mu}$ and covariance matrix $\frac{1}{N}\tilde{\Sigma}$, where $\tilde{\mu}$ is the (unknown) mean of $\tilde{g}_r$, $\tilde{\Sigma}$ is the (unknown) covariance of $\tilde{g}_r$ and N is the sample size. I.e., $f_r \sim \mathrm{N}(\tilde{\mu}, \frac{1}{N}\tilde{\Sigma})$.

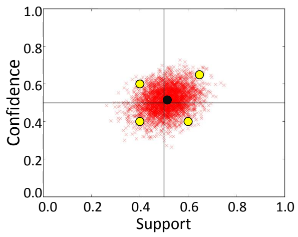
Both $\tilde{\mu}$ and $\tilde{\Sigma}$ are unknown but, using the samples for rule r, we can approximate $\tilde{\mu}$ as the *sample average* μ and $\tilde{\Sigma}$ as the *sample covariance* Σ. Now the sample mean distribution f_r may be approximated as a normal distribution with mean μ and covariance $\frac{1}{N}\Sigma$. I.e., $f_r \sim \mathrm{N}(\mu, \frac{1}{N}\Sigma)$.

EXAMPLE 4.1. *Assume that for a particular rule $\hat{r} = A \rightarrow B$, we have obtained so far the answers of 4 users:*

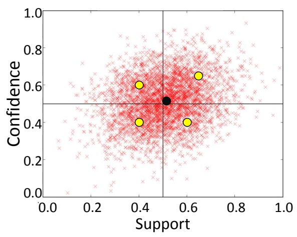
	User support	***User confidence***
User 1	0.4	0.4
User 2	0.4	0.6
User 3	0.6	0.4
User 4	0.65	0.65

The sample mean and covariance are shown in Table 1, under the column "current". The rest of the table is not necessary at this point, but will be useful as we continue with this example throughout the section. The obtained estimated Gaussian distribution of the mean $f_{\hat{r}}$ is depicted in Fig. 2(a). The dark circle is the mean, the light circles are the actual samples, and the background is darker according to where the probability mass lies.

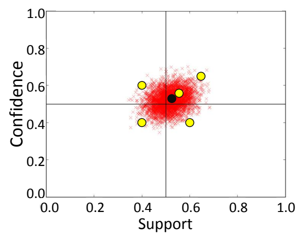
This estimation of the mean is based on the fact that, other than samples, we have no information about the mean. If we had some prior estimation for the mean distribution, this method could be extended to take it into account.



(a) $f_{\hat{r}}$



(b) $g_{\hat{r}}$



(c) $f'_{\hat{r}}$ when the next sample is $(0.55, 0.55)$

Figure 2: Distribution figures for rule $\hat{r}$ (x-axis is confidence, y-axis is support)

Estimating the sample distribution [h]. In order to choose the best next question, we need to estimate what the next answer might be for each rule. Recall that $\tilde{g}_r$ is an unknown distribution. We already know that the sample mean μ and sample covariance Σ are unbiased estimators for its mean $\tilde{\mu}$ and covariance $\tilde{\Sigma}$, respectively. Here, we choose to roughly approximate $\tilde{g}_r$ by $g_r \sim \mathrm{N}(\mu, \Sigma)$. Note that g_r cannot be precisely normal, since the support and confidence values are taken from the range $[0, 1]$. Thus, for a sample (s, c) we use the PDF $g_r((s, c) \mid s, c \in [0, 1])$ (truncated Gaussian). The notation $\mid s, c$ is omitted for brevity.

A normal distribution is a simple and general choice for describing the distribution of real-life data. However, other distributions may make good choices as well. For instance, it may be the case that user either have frequent or rare headaches. This produces two groups of samples, one with high values, and another with low ones. Such a situation may be better described by a bimodal distribution rather than a normal one.

Example 4.2. *Consider again the samples from Ex. 4.1. The distribution of $g_{\hat{r}}$, computed using μ and Σ as mean and covariance, is depicted in Fig. 2(b). Compare to Fig. 2(a) to see the difference in variance w.r.t. $f_{\hat{r}}$.*

Estimating rule significance [j]. At any point of the algorithm, based on the current best estimations for the overall confidence and support, we would like to decide for every rule r whether it is significant or not. Note that the probability of the mean being above both thresholds is the *unbiased estimator* for whether the mean is above both thresholds, since for a binary random variable, the probability for 1 is also the expectation. In addition, note that as the number of samples N grows, the covariance approaches the 0 matrix; this happens since the sample covariance Σ approaches the constant, real covariance $\tilde{\Sigma}$, and we divide it by N to estimate the covariance of the mean.

Thus, to decide on rule significance, we check whether, according to f_r, the probability for the mean to be above both thresholds is greater than 0.5. This is done by testing whether $\int_{\Theta_s}^{\infty}\int_{\Theta_c}^{\infty} f_r(s, c)\ \mathrm{d}c\,\mathrm{d}s > 0.5$ (numerical integration of bivariate normal distributions is well-studied and fast algorithms exist to reach a very high precision [14]).

Example 4.3. *Let us return to our running example. Assume that $\Theta_s = \Theta_c = 0.5$. The probability that the mean is above both thresholds, computed using the formula above, is* 0.367*. This means that there is a greater probability for the overall support and confidence of $\hat{r}$ to be below one of the thresholds, and thus our algorithm decides that it is not a significant rule.*

Example 4.3 provides another important observation: although the sample mean $\mu = (0.5125, 0.5125)$ is above both thresholds, the probability of being below one of the thresholds is greater than 0.5. Thus, using the mean itself to decide whether a rule is significant may be misleading.

Estimating current quality [f]. We suggest, as an output quality measure, the *total number of errors*. Recall that the ground truth in our case is unknown, rendering exact computation impossible; instead, we compute the *expected* number of errors, according to the probability of giving an erroneous answer on each rule. Let X_i be a random variable whose value is 1 if our algorithm makes an error in classifying rule r_i. Denote the error probability for rule r_i by $P_{\mathrm{err}}(r)$, and the domain of all possible rules by $\mathcal{R}$. The expected total number of errors is given by the formula

$$\mathbb{E}\left[\sum_{r_i \in \mathcal{R}} X_i\right] = \sum_{r_i \in \mathcal{R}} \mathbb{E}\left[X_i\right] = \sum_{r_i \in \mathcal{R}} P_{\mathrm{err}}(r_i) \qquad (1)$$

Let us now develop the formula for $P_{\mathrm{err}}(r)$. In the case that our algorithm decides that a rule r is significant, the error probability is the probability for r not to be significant, i.e., $P_{\mathrm{err}}(r) = 1 - \int_{\Theta_c}^{\infty}\int_{\Theta_s}^{\infty} f_r(s, c)\ \mathrm{d}c\,\mathrm{d}s$. In the case when the algorithm decides r is not significant, the complement of this formula to 1 is used. Note that if the decision method described above is used, $P_{\mathrm{err}}(r)$ is the probability of being on the side of the thresholds with the smaller distribution mass. In particular, it is guaranteed to be 0.5 or less.

Example 4.4. *Returning to our running example, we can now easily compute the error probability. The algorithm decides that $\hat{r}$ is not significant, the error probability is the probability of being above both thresholds –* 0.367*. This also appears in Table 1.*

The computation explained above works only for rules that have at least two samples (this is necessary for f_r to be well-defined). In order to handle rules without any user samples, we allot each rule an initial set of samples by a new user $u^* \notin \mathcal{U}$. These samples can be chosen such that the initial error probability is maximal (0.5) and the covariance is large, to reflect the fact that we have no information about this rule, yet. This also means that the next user answer is likely to reduce the error probability and covariance, i.e., our estimation becomes more accurate, as we would expect when knowledge is gained. As we obtain more user samples, the effect of these initial samples becomes negligible.

Estimating next quality [g]. We next want to estimate the effect of the next question on the total number of errors. First, we need to know how an additional sample may affect the error probability of rule r. Recall that we have estimated the sample distribution as g_r. Given one more sample from g_r in addition to the collected user samples, we are able to compute a new sample mean μ', a new sample covariance Σ', and based on these two – a new distribution of the mean f'_r. Since every sample may result in a different f'_r, we aim at finding the *expected* error $\mathbb{E}\left[P'_{\text{err}}(r)\right]$, over all possible next samples. We split the computation into two cases, depending on the probability of $\tilde{\mu}$ to be above both thresholds given the new average μ'. Formally, let s and c be the support and confidence of the next sample, drawn from g_r, and f'_r the new sample mean distribution based on them. Denote by $\varphi(s,c)$ the area of the next sample values for which r is estimated to be significant. Then,

$$\mathbb{E}\left[P'_{\text{err}}(r)\right] = \iint\limits_{\varphi(s,c)} g_r(s,c) \cdot \left[1 - \int_{\Theta_c}^{\infty}\int_{\Theta_s}^{\infty} f'_r(s',c')\, \mathrm{d}c'\mathrm{d}s'\right] \mathrm{d}c\,\mathrm{d}s$$
$$+ \iint\limits_{\neg\varphi(s,c)} g_r(s,c) \cdot \left[\int_{\Theta_c}^{\infty}\int_{\Theta_s}^{\infty} f'_r(s',c')\, \mathrm{d}c'\mathrm{d}s'\right] \mathrm{d}c\,\mathrm{d}s$$

The first summand refers to the error probability if the rule is significant after the next sample. In the outer integral we integrate over all relevant values of the next sample, and in the inner integral we compute the error probability as the chance of the mean not being above both thresholds. We compute the average of such error probabilities, weighted according to the probability of the particular next sample. Similarly, the second summand refers to the error probability if the rule is *not* significant after the next sample.

This integral cannot be computed directly, and thus we use a numerical Monte Carlo method to estimate its value [18]. This is done by repeatedly sampling s, c from g_r. For each such sample, we add it as the ith sample to the $i-1$ samples obtained from the users. From this set of i samples we compute a new sample mean and sample covariance, to find the new distribution f'_r. Now we can compute the error probability of f'_r in the exact same manner as we did with f_r above. Finally, after taking enough samples, we can average the error probabilities computed for each sample and obtain the estimator for $\mathbb{E}\left[P'_{\text{err}}(r)\right]$.

This estimator can then be plugged into formula 1 to compute the expected next total number of errors. We are now in fact computing the expectation of the expected total number of errors $\mathbb{E}\left[\mathbb{E}\left[\sum_{r_i\in\mathcal{R}} X_i\right]\right] = \sum_{r_i\in\mathcal{R}} \mathbb{E}\left[\mathbb{E}\left[X_i\right]\right]$. Note that we only ask on r, and since we treat the rules as independent, the answer only impacts r's error probability. Thus, if $r_i = r$, then $\mathbb{E}\left[\mathbb{E}\left[X_i\right]\right] = \mathbb{E}\left[P'_{\text{err}}(r)\right]$, and $\mathbb{E}\left[\mathbb{E}\left[X_i\right]\right] = P_{\text{err}}(r_i)$ otherwise.

EXAMPLE 4.5. *Returning to our running example, we examine the effect of an additional sample on the mean distribution. Assume that the fifth sample is* $(0.55, 0.55)$. *The new sample mean and covariance would be*

$$\mu' = (0.52, 0.52), \quad \Sigma' = 10^{-3} \cdot \begin{pmatrix} 2.64 & 0.65 \\ 0.65 & 2.64 \end{pmatrix}$$

Fig. 2(c) depicts the new distribution $f'_{\hat{r}}$*, where notably the covariance decreases w.r.t.* $f_{\hat{r}}$ *(Fig. 2(a)). The expected next error,* 0.332 *is lower than the current one,* 0.367.

Final ranking of the rules [e]. Given the estimations discussed thus far, we want to rank the association rules. The grade used for ranking a rule r should reflect the *expected effect* of an answer about r, on some global output quality measure, in our case the total number of errors. The chosen closed question is about the highest-ranked rule.

We explained in the previous two sections how to estimate the current error probability for a rule r and the expected error probability after one more question. Based on these two values, we can compute the expected error probability decrease resulting from one question on r, as $\mathit{grade}(r) = \mathbb{E}\left[P_{\text{err}}(r) - P'_{\text{err}}(r)\right] = P_{\text{err}}(r) - \mathbb{E}\left[P'_{\text{err}}(r)\right]$. This error decrease also reflects the effect on the total number of errors, since the only summand in formula 1 that changes when we ask about r is $P_{\text{err}}(r)$, which is replaced with the expected value of $P'_{\text{err}}(r)$.

The total number of errors has practical advantages as the quality measure. Given the current and next error probabilities for every rule, the effect on this measure is easy to compute. Consider an alternative quality measure for the output. In search algorithms such as ours, two types of quality measures are typically noted: *precision*, or the ratio of relevant results within the output, and *recall*, or the ratio of output data within the relevant data. Since there is typically a tradeoff between the two, it is common for algorithms to optimize a measure that balances between them. The standard such measure is the F-measure, defined as $2 \cdot \frac{\text{precision}\cdot\text{recall}}{\text{precision}+\text{recall}}$.[3] One may choose the question to ask based on the answer's expected effect on the F-measure. We found this approach less practical, since precision and recall can only be estimated very roughly, when the ground truth is unknown. Moreover, due to computation complexity, significant numerical errors may be encountered. In tests, while the general trends were similar, using the former approach provided higher-quality results, also w.r.t. the F-measure. See Section 6.

5. OTHER COMPONENTS

We next describe implementations for the two remaining black-boxes that are independent of ranking.

Choosing the enabled rules [d]. The following challenge is classic in association rule learning. Making computations (e.g., error estimations) for the entire domain of rules is infeasible, since it is huge. We would like to consider only a relatively small subset of rules, that are more likely to be significant than others, at each point of the algorithm. We refer to these rules as *enabled*. A closed question may only be asked on an enabled rule.

Classic data mining algorithms rely on different strategies for choosing the order in which rules are considered. Denote by *k-rule* a rule $A \to B$ such that $|A \cup B| = k$. Apriori and following works consider first only 1-rules, then only 2-rules, etc [2]. This relies on the fact that according to the definition of support, if a $k+1$-rule has a high support, all its k-sub-rules (made of subsets of its items) also have high support values. Other data mining algorithms may "jump" from k to $k+k'$, where $k' > 1$ [3]. While these jumps lead to rules that are less likely to be significant, if they turn out to be significant, we gain information not only on the considered rule but also on all of its sub-rules .

The unique settings of crowd mining prevents us from using existing strategies as-is. First, at any point we never gain "enough" information about significant k-rules. The reason is the approximate nature of our algorithm, which

[3]In our experiments in Section 6, we use the F-measure to test the performance of our algorithm.

always involves some degree of uncertainty. Thus, "exhaustive" approaches such as Apriori cannot be used. Second, the ability of people to remember large rules is limited, which calls to a focus on smaller rules. Last, we treat the rules as independent in our estimations. This renders jumps rather useless, as they increase the risk of wasting a question, without the benefit of gaining more information. How to utilize rule dependencies in our settings is left for future work.

We suggest the following variation for the order of considered rules, designed for our settings as described above. Note that unlike Apriori, the set of enabled rules at each point contains rules of different sizes.

1. We enable smaller rules first, starting from all 1-rules (using discovered items).
2. In Apriori, a $k+1$-rule is considered only if *all* its k-sub-rules have a high support. In crowd mining, rather than waiting for all significant k-rules to be known, we enable a $k+1$-rule as soon as *two* of its k-sub-rules are enabled, and have a high support. We need at least two rules in order to "cover" the $k+1$ items.
3. No jumps are taken.

This strategy is rather simple, but performs well for our settings, see Section 6. More sophisticated strategies will be considered in future work.

Deciding open vs closed questions [b]. So far, we have only focused on asking closed questions. However, as explained in Section 3, open questions are also a vital part of crowdsourcing. One reason is that open questions allow discovering new items from $\mathcal{I}$. Another pragmatic reason is that people tend to spontaneously recall only prominent rules – with very low or very high support and confidence (for instance, "When I have a stomach ache, I *almost always* drink tea"). Thus, an answer to an open question may improve our recall, by discovering an unknown significant rule.

On the other hand, open questions are closer to random samples, since the rule in the answer is unknown. As we show in Experiment 4 in Section 6, asking only open questions results in low precision and does not measure up to algorithms with closed questions. This is true even when we assume that the user answers always regard significant rules.

The decision between open and closed questions relates to the general, well-studied, problem of *exploration vs. exploitation* in sequential sampling [24]. The tradeoff is between gaining more information (in our case, finding where uncertainty exists), and utilizing existing information (in our case, reducing the uncertainty). To optimally balance the two, one must make specific assumptions on the answer distribution. Since we do not make such assumptions, we experiment with two general strategies of the exploration-exploitation paradigm, which provide approximate solutions [24].

One possible strategy, called the *ε-greedy strategy* is choosing a constant proportion of the questions to be open. This can be done by *flipping a weighted coin* with probability $1-\varepsilon$ at each iteration. This option is the simplest one, and balances the number of closed and open questions. A more sophisticated strategy, called the *ε-decreasing strategy*, relies on the fact that as we gain more information, less exploration is required. The weight for choosing an open question is initially high, and gradually decreases as more knowledge is collected. In Section 6 we compare the performance of the two strategies and show that the simple approach performs just as well as the more sophisticated one. This is also in accordance with the empirical studies in [24].

6. EXPERIMENTS

We introduce *CrowdMiner*, a system for mining association rules from the crowd. The *CrowdMiner* system is implemented in Python using an SQLite database. External libraries used were StatsModels[4], containing an implementation of multivariate Gaussian integration through MC sampling [14], and the Orange data mining library [9]. Experiments were executed on a Lenovo ThinkPad with 8GB of RAM and a 2.80 GHz Intel Core i7 processor, running Windows 7 x64. All results are averaged over 5 executions per algorithm for accuracy.

6.1 Benchmark Datasets

Since the concept of mining rules from the crowd is a new one, no benchmark datasets currently exist in the form of rules assigned to users. We propose 3 benchmark datasets upon which we run our experiments. The benchmarks are constructed in the following way.

To simulate the real world, we begin with a transaction dataset. Each user is assigned a set of transactions, which are then translated into personal association rules using the standard Apriori algorithm. Each user u now has a set of rules R_u which represent the user's "world", and their respective confidence and support values; these values are stored in what we refer to as the *real-world database.* They may then be queried as part of the crowd-mining process.

Recall that Apriori itself uses thresholds in order to decide which rules to generate. We therefore supply the Apriori algorithm with support and confidence thresholds, *lower than the thresholds used to determine rule significance.* We refer to these thresholds as the *Apriori thresholds.* The support threshold for each user was set to $\frac{2}{\#transactions}$, to ensure that each rule is supported by at least two different transactions. The confidence threshold may vary per dataset, but was generally set to 0.1 to ensure a database of manageable size.

Given the users' individual rule sets, we simulate the answers to questions as follows. When a user u is asked a closed question about some rule $r \in R_u$, she returns the support and confidence $supp_u(r)$, $conf_u(r)$. If u is asked about a rule $r \notin R_u$, she returns "don't-know", a default value having low support and confidence. A user u who is asked an open question returns one of her k-best rules, sorted first by support and then by confidence (k was usually set to 50). This is because support is generally the dominant factor when deciding whether a rule is significant or not.

The support and confidence thresholds are determined based on the data distribution so as to ensure an adequate number of significant rules. If the threshold is set too low, the difference between a significant and a non-significant rule may be negligible; if the threshold is set too high, we may not obtain enough significant rules.

Synthetic Dataset. This dataset contains transactions that were generated using the IBM Quest Data Generator [2]. We have generated three types of transactions, such that the items in the transactions of each type were selected from a domain of a different size: 200K transactions over a domain of 20 items, 100K transactions over 100 items and 50K transactions over 500 items. By combining the 3 types, we

[4]http://statsmodels.sourceforge.net/

ensured the existence of rules with low, medium and high support and confidence. This transaction set was then uniformly assigned to a set of 1,000 users. The resulting dataset is sparse, but well distributed: some rules are common to almost all the users, but most of the rules are known to only a few. We experimented with different subsets of these users, of sizes between 100 and 1,000 (over the same transaction distribution). Since the results did not change significantly, we present the results for the entire set of 1,000 users.

Retail Dataset. The retail dataset was constructed from a list of transactions representing purchases at a retail store [6]. This real-world dataset is available online, and used for benchmarking in many data mining applications. The dataset reflects the transactions of approximately 3,000 users. Since the data was anonymized, transactions were uniformly assigned to a set of users.

Wikipedia Dataset. The third dataset was extracted from Wikipedia and contains a list of real Wikipedia users and the categories they edited (each transaction is an edit session, with the categories of edited articles). We chose Simple English Wikipedia[5] because of its smaller scale, making the extraction of contributors for every article more manageable. However, the obtained database was overly specific and noisy, containing multiple categories representing the same concept (such as "movie" and "film"), and required cleaning. To do this, we made use of YAGO [21], a massive semantic taxonomy constructed from Wikipedia. First, we mapped Simple English Wikipedia articles to YAGO entities. We then used the YAGO taxonomy to "climb up" the category hierarchy, selecting for each category in each article the relevant high-level categories. This provided cleaner data with a greater degree of similarity between users, enabling meaningful rule mining. This database contains roughly 600 users, all of whom have at least one association rule.

6.2 Experimental results

Algorithms. We compare three different algorithms, each of which selects the queries in a different way:

- *Random* is a naïve algorithm that asks at each point about a random rule in the known database.
- *Greedy* is a simple algorithm that asks about the rule with the *fewest samples* at each point.
- *CrowdMiner* uses all the techniques detailed in Sections 4 and 5. At each point, it selects the question that maximizes our grade function.

The output of each algorithm is a list of its significant rules: rules which satisfy the threshold with probability > 0.5. Each experiment below compares the performance of these algorithms over time, given different scenarios.

Metrics. We evaluate our experiments using two metrics. *Errors* represents the absolute number of errors made by each algorithm: it is calculated as the sum of false-positives and false-negatives. *F-measure* is the harmonic mean of precision and recall, used to characterize the results more accurately. Naturally, we also measured precision and recall for all the experiments; some of the graphs are not shown for lack of space. All metrics were re-measured after every iteration of the algorithm. Combining all results provided us with a progression chart, showing how the F-measure / number of errors progressed over time in the execution of the algorithm. Every test used 2,000 crowd questions.

We now present the results of our empirical evaluation.

Experiment 1: Zero-Knowledge Mining. This experiment tests the performance of the 3 algorithms over a dataset for which they have no prior information: i.e., they begin with an empty sample database, and fill it by querying the real-world database. This experiment allows us to examine how accurately the algorithms mine association rules while simultaneously constructing a database.

Figure 3 shows the results for this experiment over the three benchmark datasets, for both F-measure and error metrics. Note that the error and F-measure graphs behave in a symmetric fashion (when the F-measure is high, the error number is low), and therefore in future experiments we show only the F-measure. During the execution of the algorithms, open questions and newly enabled rules increase the number of rules in the database, while closed questions improve the knowledge about existing rules. As the experiments show, the noted improvement is derived primarily from recall obtained from the enabling of larger rules: our algorithm is able to extract the largest number of significant rules. The precision differences are somewhat smaller. An example of precision–recall breakdown is shown in Figure 3(g) and 3(h) respectively. The Wikipedia dataset (Figure 3(f)) exhibits lower F-measure levels for all algorithms, since this data is highly distributed and there are many rules that apply only to small groups or even single users. Even so, *CrowdMiner* outperforms the other two algorithms.

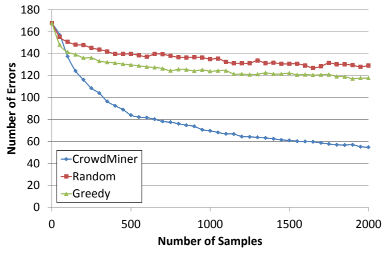
Experiment 2: Mining Over Known Item-Domains. In contrast to the previous scenario, we now examine a scenario in which the domain of items is given in advance. This simulates a case where, for example, there exists a large catalog listing all the items available for purchase in Chicago, while no information exists about consumer habits. The execution of the algorithms thus starts with a database where all the items in the domain exist as 1-rules, having initial "don't know" values.

Figure 4 shows the results for this experiment over the three benchmarks. These results are very similar to the zero-knowledge experiment, but it appears that in the first few crowd questions, the greedy algorithm benefits from the additional information provided by the catalog. Since initially all the 1-rules are in the same state, asking about the rule with the least samples is pretty much the best that can be done, and the precision of the greedy algorithm is relatively high. However, it is soon overtaken by *CrowdMiner*, which discovers more rules by enabling larger rules with a high potential for being significant.

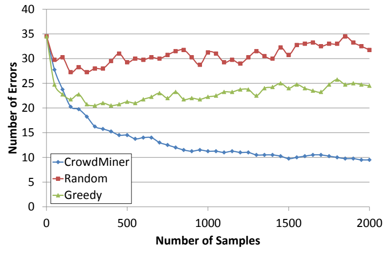
Experiment 3: Refinement for a Given Set of Rules. In this experiment, we are interested in isolating and examining the performance of the *closed question functionality* in *CrowdMiner* vs. the two competitor algorithms. All the algorithms are given an initial, randomly selected set of 400 rules. They then ask a series of closed questions only. For lack of space, we only show results for the synthetic dataset.

Figure 4 shows the results for this experiment in two metrics: (i) F-measure and (ii) errors. Since there are no open questions and no enabling of larger rules, the scope of the recall is limited.This means that the change in the F-measure graph is mainly caused by the change in precision.

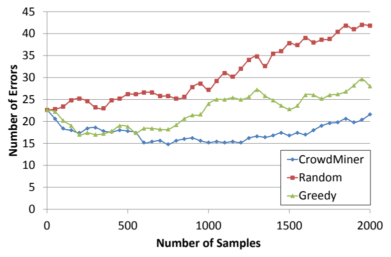
[5]`http://simple.wikipedia.org/`



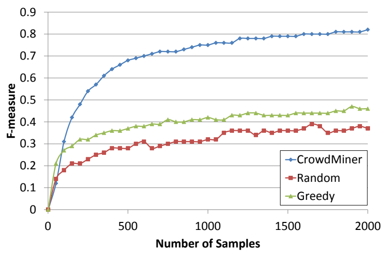
(a) Synthetic: Errors



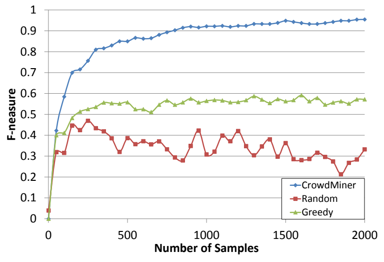
(b) Retail: Errors



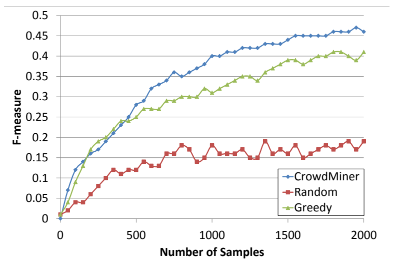
(c) Wikipedia: Errors



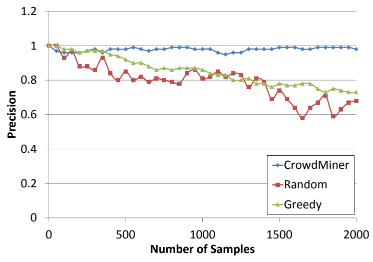
(d) Synthetic: F-measure



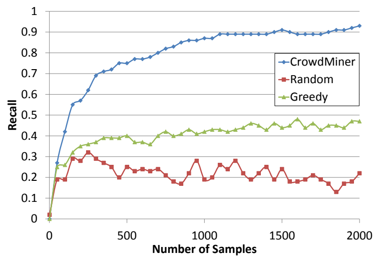
(e) Retail: F-measure



(f) Wikipedia: F-measure

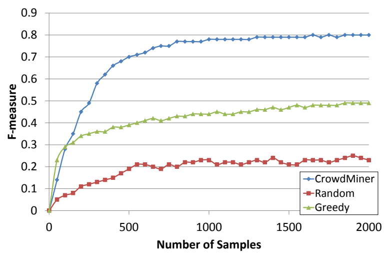


(g) Retail: Precision

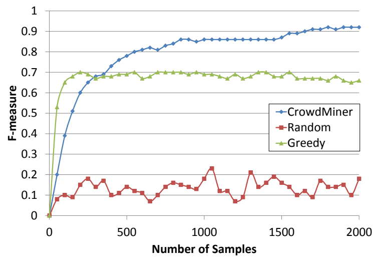


(h) Retail: Recall

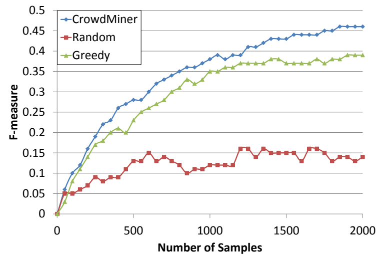
Figure 3: Zero-knowledge rule mining



(a) Synthetic: F-measure

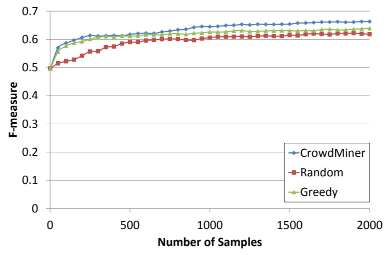


(b) Retail: F-measure

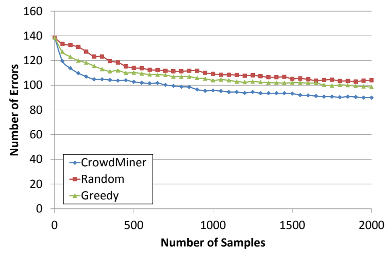


(c) Wikipedia: F-measure

Figure 4: Rule mining for known items

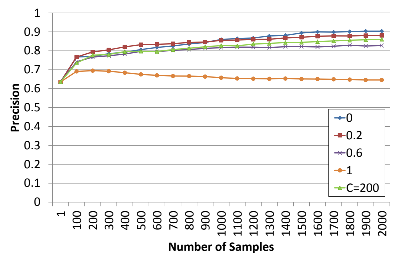


(a) Synthetic: F-measure

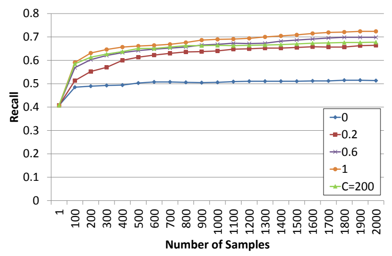


(b) Synthetic: Errors

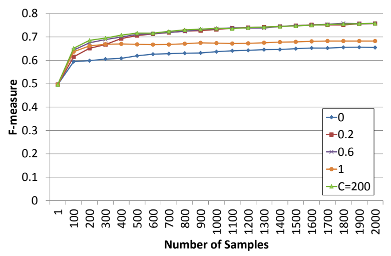
Figure 5: Rule mining over initial sample



(a) Precision



(b) Recall



(c) F-measure

Figure 6: Determining the coin flip value

The advantage of *CrowdMiner* here is statistically significant, and is preserved across additional experiments that were run using different initial sample sizes.

Experiment 4: Choosing Open or Closed Questions. As described in Section 5, we study two strategies of deciding between open and closed questions, and their effect on the performance of *CrowdMiner*. Both strategies rely on flipping a weighted coin, where the weight represents the probability of asking an open question.

The first strategy, ε-*greedy*, assigns the coin a constant weight which we range from 0 (ask only closed questions) to 1 (ask only open questions). In effect, a coin flip of 0 cannot introduce any new information into the system, while a coin flip of 1 is equivalent to sampling rules at random from users. The second strategy, ε-*decreasing* reduces the coin flip weight as a function of time: $weight(t) = \frac{C}{C+t}$. This function was run for different values of C. The first 4 values in the legends of Figure 6 represent the results for particular ε-greedy coin flip values. The last represents the best results for ε-decreasing, which were obtained for $C = 200$.

We use an initial database as in Experiment 3.

Figure 6(a) shows the precision of the results, for different weights of the coin flip. The highest precision of ε-greedy is achieved when the weight is 0, and *CrowdMiner* asks only closed questions. This makes sense, since asking closed questions leads to higher confidence within the set of known rules, and in particular within the rules estimated to be significant. In general, the higher the weight is, the lower the precision is. Figure 6(b) shows the recall of the results. Note that the highest recall is achieved when the weight is 1 (*CrowdMiner* asks only open questions), and the lowest when the weight is 0. However, note that there is a significant gap between the recall for a weight of 0 (no new information is introduced into the system) and the recall for a weight of 0.2 (a small amount of new information is introduced into the system). We also observe that a varying coin flip value does not yield better results than a fixed value. The best results, for $C = 200$, were similar to those of a fixed weight of 0.6.

We set the default weight of the coin flip to a fixed value of 0.1 for all the other experiments. This is done in order to reduce randomness in the execution of *CrowdMiner*, while still allowing enough exploration of rules and items.

7. RELATED WORK

Data procurement is one of the most important and challenging aspects of crowdsourcing [12]. Some recent works (e.g., [11, 13, 16, 19]) suggest the construction of database platforms where some components of the database (e.g., values, query parts) are crowdsourced. In such platforms, the choice of crowd questions by the query plan has to take into account various aspects such as the monetary cost, latency, accuracy, etc. For the same reasons, in the current work we attempt to maximize the accuracy of the output while asking a minimal number of crowd questions. The works of [4, 10] deal with deciding which user should be asked what questions, with the particular goal of minimizing uncertainty. However, the settings for crowd mining are more complicated: we do not know in advance what knowledge we are looking for, and in particular, which questions may be asked, or what the answers might be (for open questions).

To support the collection of open-world data, several platforms (e.g., Wikipedia) allow users to decide what data to contribute. Such platforms may collect an unbounded amount of valuable data, solely depending on contributor efforts. In contrast, our algorithm poses targeted questions, which help users recall information they may have not provided spontaneously. The knowledge gain maximization at each question leads, in particular, to user effort minimization.

Our motivation for turning to the crowd is similar to that of surveys. In surveys, the set of questions is usually small, and is *chosen in advance*, typically by experts (e.g., [7]). However, in an open world, it may be hard to know what to ask in advance. We deal with this difficulty by dynamically constructing questions based on collected knowledge.

This work has strong connections with *association rule learning*. As we noted, some principles of our algorithm are inspired by the Apriori algorithm [2]. A particularly relevant line of works [23, 27] considers cases when accessing the entire database is impossible, and learning must be done based on *data samples*. While they sample transactions, such information is not available in our settings at all. One could sample rules, which corresponds to asking only open questions. However, this allows much less control over the collected information. Our experiments (where the flip coin weight is 1, see Sec. 6) show that this approach performs worse than ours, and in particular leads to a low precision.

The goal in *active learning* [15], another related research field, is incrementally choosing data items that should be labeled by an expert, in a manner that maximizes the knowledge gain. Some works in this field are focused on scenarios with human experts, whose labels may be erroneous [8, 20, 26]. They address this problem by trying to identify preferred experts, or to filter the noise (error rate) in the answers. In contrast, we try to learn about the distribution of user answers, and there are no "wrong" answers. An interesting research direction would be utilizing techniques from these works for identifying, e.g., "representative" users, whose personal rules tend to reflect the population average. Giving more weight to these users may improve our algorithm's learning curve, but is non-trivial as explained in Section 8.

The well-studied multi-armed bandit [24] problem is: given a set of slot-machines, which should be played at each step to maximize reward? In Sections 5 and 6.2 we examine strategies for balancing exploration and exploitation, developed for this kind of problems.

8. CONCLUSION AND EXTENSIONS

This work studies, for the first time, mining association rules from the crowd, in common scenarios where transactions are not available. We have defined a formal model for crowd mining, and identified the differences from the classic data mining paradigm; presented the generic components of an effective crowd-mining framework; suggested an implementation for these components, including error probability estimation where the ground truth is unknown; and conducted an experimental study with the prototype system *CrowdMiner*, proving that our algorithm outperforms baseline competitor algorithms for different data benchmarks. In particular, it outperforms an algorithm which relies solely on rule sampling (via open questions). In our experiments we also show the importance of cleverly choosing closed questions, and gradually enabling larger rules.

We note that our algorithm, as presented in the paper, can be extended in a natural manner from choosing a single question to choosing the *best bulk of next K questions*, by means of dynamic programming. Such an extension may improve the latency of the algorithm, since it allows posing the K questions in parallel. Details about the required adjustments to the error estimations and the algorithm are omitted because of lack of space.

One possible optimization for our algorithm is by taking into account the *dependencies* between rules when interpreting user answers. In this manner, more knowledge may be extracted and utilized from each answer.

We have chosen the aggregation function to be the *average*, which gives all the users equal weights. For some applications, however, one may want to give more weight to users who are trusted, or have more underlying transactions (e.g., users that practice folk medicine more often). Extending our results to this case is not trivial, since when the distribution of weights is not known in advance, it is hard to predict the effect of a user answer. We leave this for future research.

Finally, our framework consists of many black-boxes. We presented one promising implementation for them, but it is interesting to study other implementations, which use prior knowledge on the data distribution, the users, etc.

Acknowledgments. This work has been partially funded by the European Research Council under the FP7, ERC grants Webdam, agreement 226513, and MoDaS, agreement 291071, and by the Israel Ministry of Science.

9. REFERENCES

[1] R. Agrawal, T. Imieliński, and A. Swami. Mining association rules between sets of items in large databases. *SIGMOD Record*, 22(2), 1993.
[2] R. Agrawal and R. Srikant. Fast algorithms for mining association rules. In *VLDB*, 1994.
[3] R. J. Bayardo Jr. Efficiently mining long patterns from databases. *SIGMOD Record*, 27(2), 1998.
[4] R. Boim, O. Greenshpan, T. Milo, S. Novgorodov, N. Polyzotis, and W. Tan. Asking the right questions in crowd data sourcing. In *ICDE*, 2012.
[5] N. Bradburn, L. Rips, and S. Shevell. Answering autobiographical questions: The impact of memory and inference on surveys. *Science*, 236(4798), 1987.
[6] T. Brijs, G. Swinnen, K. Vanhoof, and G. Wets. Using association rules for product assortment decisions: A case study. In *KDD*, 1999.
[7] J. M. Converse and S. Presser. *Survey questions: Handcrafting the standardized questionnaire.* Number 63 in Quantitative Applications in Social Sciences. Sage Publications Inc., 1986.
[8] O. Dekel and O. Shamir. Vox populi: Collecting high-quality labels from a crowd. In *COLT*, 2009.
[9] J. Demšar, B. Zupan, G. Leban, and T. Curk. Orange: From experimental machine learning to interactive data mining. *PKDD*, 2004.
[10] D. Deutch, O. Greenshpan, B. Kostenko, and T. Milo. Declarative platform for data sourcing games. In *WWW*, 2012.
[11] A. Doan, M. Franklin, D. Kossmann, and T. Kraska. Crowdsourcing applications and platforms: A data management perspective. *PVLDB*, 4(12), 2011.
[12] A. Doan, R. Ramakrishnan, and A. Y. Halevy. Crowdsourcing systems on the World-Wide Web. *Commun. ACM*, 54(4), 2011.
[13] M. Franklin, D. Kossmann, T. Kraska, S. Ramesh, and R. Xin. CrowdDB: answering queries with crowdsourcing. In *SIGMOD*, 2011.
[14] A. Genz. Numerical computation of rectangular bivariate and trivariate normal and t probabilities. *Statistics and Computing*, 14, 2004.
[15] M. Lindenbaum, S. Markovitch, and D. Rusakov. Selective sampling for nearest neighbor classifiers. *Machine Learning*, 54(2), 2004.
[16] A. Marcus, E. Wu, D. Karger, S. Madden, and R. Miller. Crowdsourced databases: Query processing with people. In *CIDR*, 2011.
[17] Amazon's Mechanical Turk. `https://www.mturk.com`.
[18] M. Newman and G. Barkema. *Monte Carlo methods in statistical physics.* Oxford University Press, 1999.
[19] A. G. Parameswaran and N. Polyzotis. Answering queries using humans, algorithms and databases. In *CIDR*, 2011.
[20] V. Sheng, F. Provost, and P. Ipeirotis. Get another label? Improving data quality and data mining using multiple, noisy labelers. In *SIGKDD*, 2008.
[21] F. M. Suchanek, G. Kasneci, and G. Weikum. Yago: A core of semantic knowledge. In *WWW*, 2007.
[22] J. Surowiecki. The wisdom of crowds: Why the many are smarter than the few and how collective wisdom shapes business. *Economies, Societies and Nations*, 2004.
[23] H. Toivonen. Sampling large databases for association rules. In *VLDB*, 1996.
[24] J. Vermorel and M. Mohri. Multi-armed bandit algorithms and empirical evaluation. *ECML*, 2005.
[25] L. von Ahn and L. Dabbish. Designing games with a purpose. *Commun. ACM*, 51(8), 2008.
[26] Y. Yan, R. Rosales, G. Fung, and J. Dy. Active learning from crowds. In *ICML*, 2011.
[27] M. Zaki, S. Parthasarathy, W. Li, and M. Ogihara. Evaluation of sampling for data mining of association rules. In *RIDE*, 1997.